HIGH PERFORMANCE COMPUTING AND GRIDS IN ACTION

Advances in Parallel Computing

This book series publishes research and development results on all aspects of parallel computing. Topics may include one or more of the following: high-speed computing architectures (Grids, clusters, Service Oriented Architectures, etc.), network technology, performance measurement, system software, middleware, algorithm design, development tools, software engineering, services and applications.

Series Editor:
Professor Dr. Gerhard R. Joubert

Volume 16

Recently published in this series

Vol. 15. C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr and F. Peters (Eds.), Parallel Computing: Architectures, Algorithms and Applications

Volumes 1–14 published by Elsevier Science.

ISSN 0927-5452

High Performance Computing and Grids in Action

Edited by

Lucio Grandinetti

Center of Excellence on High Performance Computing,
University of Calabria, Italy

IOS Press

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

ISBN 978-1-58603-839-7
Library of Congress Control Number: 2008921029

Publisher
IOS Press
Nieuwe Hemweg 6B
1013 BG Amsterdam
Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

Distributor in the UK and Ireland
Gazelle Books Services Ltd.
White Cross Mills
Hightown
Lancaster LA1 4XS
United Kingdom
fax: +44 1524 63232
e-mail: sales@gazellebooks.co.uk

Distributor in the USA and Canada
IOS Press, Inc.
4502 Rachael Manor Drive
Fairfax, VA 22032
USA
fax: +1 703 323 3668
e-mail: iosbooks@iospress.com

PRINTED IN THE NETHERLANDS

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Foreword

The book series *Advances in Parallel Computing* was launched in 1988. The first volume was published by Elsevier Science in September 1989. Until 2005 a total of fourteen volumes were published. These comprised conference and workshop proceedings as well as publications covering particular topics.

Since the beginning of 2007 the publication of the book series is continued by IOS Press. The aims and scope remain as before. Thus the book series publishes material covering all aspects of parallel and high speed computing.

Books published in the series can include material from conferences and workshops covering all aspects of parallel computing. This includes architectures, high speed networks, algorithms and software as well as any particular topics on high performance processing to solve real-life problems in all areas, including scientific, engineering and multidisciplinary applications.

In order to maintain the high quality standard of the book series only refereed material is accepted for publication.

Two new volumes have already been accepted and were completed during 2007. These will appear in print early in 2008. The original numbering of the series is continued, thus these volumes will be numbered 15 and 16 respectively.

This sixteenth volume titled *High Performance Computing and Grids in Action*, edited by Lucio Grandinetti, is an example of the high quality aimed at with the book series. The book comprises contributions by a number of the most well-known authors in the field of Grid computing and is a milestone publication in this field.

Applications for the publication of material in *Advances in Parallel Computing* can be submitted to the Series Editor or IOS Press.

Gerhard R. Joubert
Series Editor

This page intentionally left blank

Editor's Preface

Advancement of Science and Technology and its impact on the real life applications is more and more strictly related to the progress and availability of high performance parallel computer systems and Grids, the novel networked infrastructures aimed to organize and optimize the use of a huge amount of distributed data processing and computing resources.

The book collects in four chapters a selection of papers presented at the 2006 International Advanced Research Workshop on High Performance Computing and Grids in Cetraro, Italy; these papers are related to some fundamental advances in parallel computer systems and their future developments and to the establishment and evolution of Grids fundamentals, implementation, deployment.

The aim is to cover different points of view in the field by actors playing different roles, to orchestrate and correlate their interconnection and coherence, and - above all - to show behaviours, impacts, performances of architectures, systems, services and organizations in action.

Accordingly the expected audience would be broad, mainly made up by computer scientists, graduate students, post doc researchers, specialists of computing and data centers, computer engineers and architects, project leaders, information system planners, professional technologists.

The book structure will help the reader to accomplish the objective mentioned above. An introductory monograph by I. Foster discusses the general and appealing issue on when and how to advance the state of the art in scientific software and infrastructure. Chapters 1 and 2 deal with some fundamental topics on High Performance Computing and Grids. Chapter 3 surveys different aspects of Grid Computing use, in particular tools and services for making available and deployable grids. Chapter 4 deals with applications; these in general are an indispensable motivation for researching and developing new fields of science and related technologies.

It is my pleasure thanking and acknowledging the contribution of many individuals that have made this book possible.

Gerhard Joubert, the Editor of the IOS Press book series "Advances in Parallel Computing" which now includes this volume, for his support, encouragement and advice on editorial issues and for writing the Foreword.

Jack Dongarra, Ian Foster, Geoffrey Fox, Miron Livny, among others, for advising on scientific issues before, during and after the workshop organization.

My colleagues from the Center of Excellence on High Performance Computing and the Department of Electronics, Informatics and Systems at University of Calabria, Italy are thanked for their constructive comments.

I feel indebted to Maria Teresa Guaglianone for her dedication in handling the book's material and for keeping, nicely and effectively, relationships with the contributing Authors and the Publisher.

Finally, the support of Hewlett Packard towards the book publication is acknowledged and in particular the persistent consideration and help given by Frank Baetke.

Lucio Grandinetti
Professor and Director
Center of Excellence on
High Performance Computing
University of Calabria, Italy

List of Contributors

David Abramson
Faculty of Information Technology
Monash University
Australia

William E. Allcock
University of Chicago
and Argonne National Laboratory
United States

Giovanni Aloisio
CACT/NNL
University of Salento, Lecce
and Euro-Mediterranean Centre for Climate Change
Italy

Phil Andrews
San Diego Supercomputer Center
United States

Owen Appleton
CERN
Geneva
Switzerland

Ruth Aydt
National Center for Supercomputing Applications
United States

Rosa Badia
Barcelona Supercomputing Center
Barcelona
Spain

Ray Bair
University of Chicago
and Argonne National Laboratory
United States

Henry Bal
Vrije Universiteit
The Netherlands

Natasha Balac
San Diego Supercomputer Center
United States

Bryan Banister
San Diego Supercomputer Center
United States

Trish Barker
National Center for Supercomputing Applications
United States

Mark Bartelt
California Institute of Technology
United States

Dominic Battré
TU Berlin, Complex and Distributed IT Systems
Germany

Pete Beckman
University of Chicago
and Argonne National Laboratory
United States

Francine Berman
San Diego Supercomputer Center
United States

Gary Bertoline
Purdue University
United States

Ivan Beschastnikh
Computer Science and Engineering
University of Washington
Box 352350, Seattle, WA 98195
United States

Alan Blatecky
University of North Carolina
United States

Jay Boisseau
Texas Advanced Computing Center
United States

Jim Bottum
Clemson University
United States

Sharon Brunett
California Institute of Technology
United States

Julian Bunn
California Institute of Technology
United States

Joseph Burrescia
ESnet, Lawrence Berkeley National Laboratory
United States

Michelle Butler
National Center for Supercomputing Applications
United States

Alfredo Buttari
Department of Computer Science
University of Tennessee Knoxville
United States

Massimo Cafaro
CACT/NNL
University of Salento, Lecce
and Euro-Mediterranean Centre for Climate Change
Italy

B. Barla Cambazoglu
Dept. of Biomedical Informatics
The Ohio State University
United States

Franck Cappello
INRIA
France

David Carver
Texas Advanced Computing Center
United States

Umit Catalyurek
Dept. of Biomedical Informatics
The Ohio State University
United States

Charlie Catlett
University of Chicago
and Argonne National Laboratory
United States

Barbara M. Chapman
Department of Computer Science
University of Houston
United States

Marcus Christie
School of Informatics
Indiana University, Bloomington, IN
United States

John Cobb
Oak Ridge National Laboratory
United States

Tim Cockerill
National Center for Supercomputing Applications
United States

Michael Collins
ESnet, Lawrence Berkeley National Laboratory
United States

Carmela Comito
Dept. of Electronics, Informatics and Systems
University of Calabria
Rende, CS
Italy

Peter F. Couvares
University of Wisconsin
United States

Maytal Dahan
Texas Advanced Computing Center
United States

Eli Dart
ESnet, Lawrence Berkeley National Laboratory
United States

Gargi Dasgupta
IBM India Research Laboratory
New Delhi
India

Ewa Deelman
USC Information Sciences Institute
Marina Del Rey, CA 90292
United States

Yu Deng
Community Grids Laboratory
and Department of Computer Science
Indiana University
United States

Diana Diehl
San Diego Supercomputer Center
United States

Jack Dongarra
Department of Computer Science
University of Tennessee Knoxville
and Oak Ridge National Laboratory
United States

Thom Dunning
National Center for Supercomputing Applications
United States

Onyeka Ezenwoye
Florida International University
Miami, FL
United States

Sandro Fiore
CACT/NNL
University of Salento, Lecce
and Euro-Mediterranean Centre for Climate Change
Italy

Liana Fong
IBM T.J. Watson Research Center
Yorktown Heights, NY
United States

Ian Foster
University of Chicago
and Argonne National Laboratory
United States

Robert J. Fowler
Renaissance Computing Institute (RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517
United States

Geoffrey Fox
Community Grids Laboratory
Department of Computer Science
and School of Informatics
Indiana University
United States

Jim Gagliardi
ESnet, Lawrence Berkeley National Laboratory
United States

Kelly Gaither
Texas Advanced Computing Center
United States

Todd Gamblin
Renaissance Computing Institute (RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517
United States

Dennis Gannon
School of Informatics
Indiana University, Bloomington, IN
United States

Sebastien Goasguen
Clemson University
United States

Sergei Gorlatch
Jens Müller-Iden
University of Münster
Germany

Wojtek James Goscinski
Faculty of Information Technology
Monash University
Australia

Lucio Grandinetti
High Performance Computing Center
University of Calabria
Rende, CS
Italy

Michael Grobe
Indiana University
and Purdue University
Indianapolis
United States

Paul Groth
School of Electronics and Computing Science
University of Southampton
United Kingdom

Chin Guok
ESnet, Lawrence Berkeley National Laboratory
United States

Metin Gurcan
Dept. of Biomedical Informatics
The Ohio State University
United States

Dave Hart
San Diego Supercomputer Center
United States

Shannon Hastings
Dept. of Biomedical Informatics
The Ohio State University
United States

Felix Heine
TU Berlin, Complex and Distributed IT Systems
Germany

Matt Heinzel
University of Chicago
and Argonne National Laboratory
United States

Chris Hempel
Texas Advanced Computing Center
United States

Howard Ho
IBM Almaden Research Center
San Jose, CA
United States

André Höing
TU Berlin, Complex and Distributed IT Systems
Germany

Yi Huang
School of Informatics
Indiana University, Bloomington, IN
United States

Wendy Huntoon
Pittsburgh Supercomputing Center
United States

Maria Carmen Incutti
High Performance Computing Center
University of Calabria
Rende, CS
Italy

Joseph Insley
University of Chicago
and Argonne National Laboratory
United States

Scott Jensen
School of Informatics
Indiana University, Bloomington, IN
United States

Chris Jesshope
Informatics Institute
University of Amsterdam
The Netherlands

Hai Jin
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074
China

William Johnston
ESnet, Lawrence Berkeley National Laboratory
United States

Bob Jones
CERN
Geneva
Switzerland

Christopher Jordan
San Diego Supercomputer Center
United States

Ivan Judson
University of Chicago
and Argonne National Laboratory
United States

Anke Kamrath
San Diego Supercomputer Center
United States

Gopi Kandaswamy
Renaissance Computing Institute (RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517
United States

Odej Kao
TU Berlin, Complex and Distributed IT Systems
Germany

Nicholas Karonis
Northern Illinois University
and University of Chicago
and Argonne National Laboratory
United States

Carl Kesselman
University of Southern California
Information Sciences Institute
United States

Sawsan Khuri
University of Miami
Miami, FL
United States

Jun Kong
Dept. of Biomedical Informatics
The Ohio State University
United States

Patricia Kovatch
San Diego Supercomputer Center
United States

Dieter Kranzlmüller
GUP, Joh. Kepler University
Linz
Austria

Tashin Kurc
Dept. of Biomedical Informatics
The Ohio State University
United States

Jakub Kurzak
Department of Computer Science
University of Tennessee Knoxville
United States

Lex Lane
National Center for Supercomputing Applications
United States

Stephen Langella
Dept. of Biomedical Informatics
The Ohio State University
United States

Julie Langou
Department of Computer Science
University of Tennessee Knoxville
United States

Julien Langou
University of Colorado at Denver
and Health Sciences Center
United States

Scott Lathrop
University of Chicago
and Argonne National Laboratory
United States

Erwin Laure
CERN
Geneva
Switzerland

Sangmi Lee Pallickara
School of Informatics
Indiana University, Bloomington, IN
United States

Michael Levine
Pittsburgh Supercomputing Center
United States

David Lifka
Cornell University
United States

Lee Liming
University of Chicago
and Argonne National Laboratory
United States

Ning Liu
School of Informatics
Indiana University, Bloomington, IN
United States

Yanbin Liu
IBM T.J. Watson Research Center
Yorktown Heights, NY
United States

Miron Livny
University of Wisconsin Madison
Madison, WI 53706
United States

Rich Loft
National Center for Atmospheric
Research
United States

Steve Luis
Florida International University
Miami, FL
United States

Piotr Luszczek
Department of Computer Science
University of Tennessee Knoxville
United States

Anirban Mandal
Renaissance Computing Institute
(RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517
United States

Doru Marcusiu
National Center for Supercomputing
Applications
United States

Francesco Mari
CESIC-NEC
c/o University of Calabria
Rende, CS
Italy

Suresh Marru
School of Informatics
Indiana University, Bloomington, IN
United States

Jim Marsteller
Pittsburgh Supercomputing Center
United States

Stuart Martin
University of Chicago
and Argonne National Laboratory
United States

Scott McCauley
Indiana University
United States

John McGee
University of North Carolina
United States

Laura McGinnis
Pittsburgh Supercomputing Center
United States

Michael McRobbie
Indiana University
United States

Gaurang Mehta
USC Information Sciences Institute
Marina Del Rey, CA 90292
United States

Paul Messina
University of Chicago
and Argonne National Laboratory
and California Institute of Technology
United States

Joe Metzger
ESnet, Lawrence Berkeley National Laboratory
United States

Simon Miles
Department of Computer Science
King's College London
United Kingdom

Maria Mirto
CACT/NNL
University of Salento, Lecce
and Euro-Mediterranean Centre for Climate Change
Italy

Reagan Moore
San Diego Supercomputer Center
United States

Richard Moore
San Diego Supercomputer Center
United States

Luc Moreau
School of Electronics and Computing Science
University of Southampton
United Kingdom

Jens Müller-Iden
University of Münster
Germany

Steve Munroe
School of Electronics and Computing Science
University of Southampton
United Kingdom

Suman Nadella
Computation Institute
The University of Chicago
5801 S. Ellis Avenue, Chicago, IL 60637
United States

Sivaramakrishnan Narayanan
Dept. of Biomedical Informatics
The Ohio State University
United States

J.P. Navarro
University of Chicago
and Argonne National Laboratory
United States

Alessandro Negro
CACT/NNL
University of Salento, Lecce
and Euro-Mediterranean Centre for Climate Change
Italy

Jeff Nichols
Oak Ridge National Laboratory
United States

Kevin Oberman
ESnet, Lawrence Berkeley National Laboratory
United States

Mike O'Connor
ESnet, Lawrence Berkeley National Laboratory
United States

Scott Oster
Dept. of Biomedical Informatics
The Ohio State University
United States

Tony Pan
Dept. of Biomedical Informatics
The Ohio State University
United States

Michael E. Papka
University of Chicago
and Argonne National Laboratory
United States

Andy Pavlo
University of Wisconsin Madison
Madison, WI 53706
United States

Rob Pennington
National Center for Supercomputing Applications
United States

Srinath Perera
School of Informatics
Indiana University, Bloomington, IN
United States

Marlon E. Pierce
Community Grids Laboratory
Department of Computer Science
and School of Informatics
Indiana University
United States

Greg Pike
Oak Ridge National Laboratory
United States

Beth Plale
School of Informatics
Indiana University, Bloomington, IN
United States

Jim Pool
California Institute of Technology
United States

Ruth Pordes
Fermi National Accelerator Laboratory
PO Box 500, Batavia, Illinois
United States

Allan K. Porterfield
Renaissance Computing Institute (RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517
United States

Anthony Praino
IBM T.J. Watson Research Center
Yorktown Heights, NY
United States

Jean-Pierre Prost
IBM Products and Solutions Support Center
Montpellier
France

Ahmed Radwan
University of Miami
Miami, FL
United States

Lavanya Ramakrishnan
Renaissance Computing Institute (RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517
United States

Raghu Reddy
Pittsburgh Supercomputing Center
United States

Daniel A. Reed
Renaissance Computing Institute
(RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517
United States

Tony Rimovsky
National Center for Supercomputing
Applications
United States

Eric Roberts
Texas Advanced Computing Center
United States

Ralph Roskies
Pittsburgh Supercomputing Center
United States

Seyed Masoud Sadjadi
Florida International University
Miami, FL
United States

Rizos Sakellariou
School of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
United Kingdom

Joel Saltz
Dept. of Biomedical Informatics
The Ohio State University
United States

Sergiu Sanielevici
Pittsburgh Supercomputing Center
United States

Tobias Schröter
University of Braunschweig
Germany

J. Ray Scott
Pittsburgh Supercomputing Center
United States

Olcay Sertel
Dept. of Biomedical Informatics
The Ohio State University
United States

Anurag Shankar
Indiana University
United States

Ashis Sharma
Dept. of Biomedical Informatics
The Ohio State University
United States

Mark Sheddon
San Diego Supercomputer Center
United States

Satoshi Shirisuna
School of Informatics
Indiana University, Bloomington, IN
United States

Shivkumar Shivaji
IBM Almaden Research Center
San Jose, CA
United States

Mike Showerman
National Center for Supercomputing
Applications
United States

Derek Simmel
Pittsburgh Supercomputing Center
United States

Yogesh Simmhan
School of Informatics
Indiana University, Bloomington, IN
United States

Abe Singer
San Diego Supercomputer Center
United States

Gurmeet Singh
USC Information Sciences Institute
Marina Del Rey, CA 90292
United States

Dane Skow
University of Chicago
and Argonne National Laboratory
United States

Aleksander Slominski
School of Informatics
Indiana University, Bloomington, IN
United States

Shava Smallen
San Diego Supercomputer Center
United States

Warren Smith
Texas Advanced Computing Center
United States

Carol Song
Purdue University
United States

Rick Stevens
University of Chicago
and Argonne National Laboratory
United States

Craig Stewart
Indiana University
United States

Robert B. Stock
Pittsburgh Supercomputing Center
United States

Nathan Stone
Pittsburgh Supercomputing Center
United States

Mei-Hui Su
USC Information Sciences Institute
Marina Del Rey, CA 90292
United States

Yiming Sun
School of Informatics
Indiana University, Bloomington, IN
United States

Domenico Talia
Dept. of Electronics, Informatics and Systems
University of Calabria
Rende, CS
Italy

Stanimire Tomov
Department of Computer Science
University of Tennessee Knoxville
United States

John Towns
National Center for Supercomputing Applications
United States

Nick Trebon
Computation Institute
The University of Chicago
5801 S. Ellis Avenue, Chicago, IL 60637
United States

Tomislav Urban
Texas Advanced Computing Center
United States

Salvatore Vadacca
CACT/NNL
University of Salento, Lecce
and Euro-Mediterranean Centre for Climate Change
Italy

Karan Vahi
USC Information Sciences Institute
Marina Del Rey, CA 90292
United States

Nithya Vijayakumar
School of Informatics
Indiana University, Bloomington, IN
United States

Mike Vildibill
San Diego Supercomputer Center
and Sun Microsystems
United States

Balaji Viswanathan
IBM India Research Laboratory
New Delhi
India

Edward Walker
Texas Advanced Computing Center
United States

Von Welch
National Center for Supercomputing Applications
United States

Patrick Welsh
University of North Florida
Jacksonville, FL
United States

R. Kent Wenger
University of Wisconsin Madison
Madison, WI 53706
United States

Nancy Wilkins-Diehr
San Diego Supercomputer Center
United States

Roy Williams
California Institute of Technology
United States

Linda Winkler
University of Chicago
and Argonne National Laboratory
United States

Yonghong Yan
Department of Computer Science
University of Houston
United States

Viktor Yarmolenko
School of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
United Kingdom

Akmal Younis
University of Miami
Miami, FL
United States

Huapeng Yuan
Community Grids Laboratory
and Department of Computer Science
Indiana University
United States

Qin Zhang
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074
China

Lan Zhao
Purdue University
United States

Ran Zheng
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074
China

Ann Zimmerman
University of Michigan
United States

This page intentionally left blank

Contents

Chapter 4. Grids and High Performance Computing in Action

This page intentionally left blank

Advancement of the State of the Art in Scientific Software and Infrastructure

This page intentionally left blank

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Human-Machine Symbiosis, 50 Years On

Ian FOSTER
Computation Institute, University of Chicago and Argonne National Laboratory
Mathematics and Computer Science Division, Argonne National Laboratory
Department of Computer Science, University of Chicago

Abstract. Licklider advocated in 1960 the construction of computers capable of working symbiotically with humans to address problems not easily addressed by humans working alone. Since that time, many of the advances that he envisioned have been achieved, yet the time spent by human problem solvers in mundane activities remains large. I propose here four areas in which improved tools can further advance the goal of enhancing human intellect: services, provenance, knowledge communities, and automation of problem-solving protocols.

Keywords. Licklider, man-computer symbiosis, provenance, services

Introduction

In his classic 1960 paper, *Man-Computer Symbiosis* [22], L.C.R Licklider wrote of how monitoring his time spent at work led him to discover that:

> *About 85 per cent of my "thinking" time was spent getting into a position to think, to make a decision, to learn something I needed to know. Much more time went into finding or obtaining information than into digesting it. Hours went into the plotting of graphs, and other hours into instructing an assistant how to plot. When the graphs were finished, the relations were obvious at once, but the plotting had to be done in order to make them so. At one point, it was necessary to compare six experimental determinations of a function relating speech-intelligibility to speech-to-noise ratio. No two experimenters had used the same definition or measure of speech-to-noise ratio. Several hours of calculating were required to get the data into comparable form. When they were in comparable form, it took only a few seconds to determine what I needed to know.*
>
> *Throughout the period I examined, in short, my "thinking" time was devoted mainly to activities that were essentially clerical or mechanical: searching, calculating, plotting, transforming, determining the logical or dynamic consequences of a set of assumptions or hypotheses, preparing the way for a decision or an insight. Moreover, my choices of what to attempt and what not to attempt were determined to an embarrassingly great extent by considerations of clerical feasibility, not intellectual capability*.

These observations led him to advocate the use of computers to, in essence, "augment human intellect by freeing it from mundane tasks"—a goal that Doug Engelbart would soon also pursue, with great success [8].

Almost 50 years later, we have personal computers, immensely powerful software, huge online databases [18], and a ubiquitous Internet (another Licklider idea [21]). Our intellect has indeed been augmented: we can, for example, perform computations and data comparisons in seconds that might have taken Licklider hours, days, or years.

In other respects, however, the situation is less rosy. While we could probably process Licklider's six speech datasets in seconds rather than hours, we will probably still struggle with incompatible formats, and may well be dealing with six million or even six billion objects. Meanwhile, while the advent of the Web has dramatically increased access to data, it can still be exceedingly difficult to discover relevant data and to make sense of that data once it is located. And as we automate various aspects of the problem solving process, other activities emerge as the time-consuming "mechanical" steps. For example, in biology, DNA microarrays allows ten of thousands of measurements to be performed in the time that a researcher might have previously taken to perform a single measurement [35]. However, the amount of time per day that a researcher spends in "mechanical" labor may be no less: experiments must still be set up, data collected and stored, results analyzed. In other words, there are still just as many opportunities to automate the routine and mechanical.

This discussion emphasizes that as we near the 50th anniversary of Licklider's paper, the need for man-computer symbiosis is no less urgent. However, we must demand far more from our computers than we did in 1960.

In the spirit of celebrating Licklider's legacy, I discuss here four related areas in which I believe significant progress can be made in further augmenting human intellect via the automation of the mundane and mechanical.

First, I examine how service-oriented architectures can make powerful information tools available over the network, for discovery and use by both people and programs. By permitting distribution of function, "service oriented science" (SOS) systems can both greatly reduce barriers to accessing existing intellectual tools—and permit (via the creation of networks of interacting services) the creation of new tools.

Second, I discuss how we can automate the documentation of data and computational results, so that users and programs alike can determine how much confidence to place in computational results. Such provenance mechanisms are an essential prerequisite to any serious attempt to realize SOS on a large scale.

Third, I point out how technology can facilitate the construction of effective communities, and thus increase the scale at which SOS techniques are applied and sustained.

Fourth and finally, I propose that the reach and impact of SOS, provenance, and community tools can be expanded by automating science protocols: extending the reach of automation to encompass not just simple computational tasks but also more complex procedures that may include experimental activities.

None of this material is new or rigorous. Nor is my review of the state of the art anything more than suggestive. Nevertheless, I hope that this presentation spurs some thoughts in my readers on where and how to advance the state of the art in scientific software and infrastructure.

1. Service Oriented Science

Emerging "digital observatories" provide online access to hundreds of terabytes of data in dozens of archives, via uniform interfaces [39]. These systems allow astronomers to

pose and answer in seconds, and from their desk, questions that might previously have required years of observation in remote observatories. For example, astronomers can combine data from different archives to identify faint objects that are visible in the infrared spectrum but not the optical—so called brown dwarves [40].

In order to build such systems, astronomers have defined conventions for describing the contents of data archives and for the messages used to request and receive data. Thus, clients can discover and access data from different sources without writing custom code for each specific data source. These conventions address both low-level details of the format of the messages exchanged between clients and services and higher-level details concerning message contents. Web Services [5] specifications and software are widely used to address lower-level concerns; higher-level concerns tend to be addressed by more application-specific conventions, such as the VOTable specification [28].

Codified interfaces allow not only humans but also programs to access services. Indeed, it is arguably automated access by software programs that really makes such systems significant. In the time that a human takes to locate one useful piece of information, a program may access and integrate data from many sources and identify relationships that a human would never discover unaided. Thus, we can discover brown dwarves, integrate information automatically from genome and protein sequence databases to infer metabolic pathways [29], and search environmental data for extreme events.

Not only data but also programs that operate on data can be encapsulated as services, as can sensors, numerical simulations, and programs that perform other computational tasks. Networks of such services can be constructed that perform complex computational activities with little or no human intervention. Systems that are thus structured in terms of communicating services are called *service-oriented architectures.* I use the term *service-oriented science* (SOS) [11] to refer to scientific research assisted or performed by such distributed networks of interoperating services.

Many believe that SOS methods are vital for dealing with the rapidly growing volume of scientific data and the increasing complexity of scientific computing and research. In principle, SOS methods make it possible to decompose and distribute responsibility for complex tasks, so that many members of a community can participate in the construction of an eventual solution.

The successful realization of SOS is not simply a question of using Web Services or similar technologies to encapsulate data and software. We also need:

- **Resources** (data, software, sensors, etc.) that are viewed as valuable by multiple people, and reward systems that motivate people to construct and operate services that provide access to those resources. These "reward systems" can range from payment to peer approval and professional advancement.
- Supporting **software and services** that allow clients to discover services, determine whether services meet their needs, and make sense of results returned by services. Depending on context, these mechanisms can range from simple natural language descriptions of service capabilities and contents to sophisticated metadata, constructed according to agreed-upon ontologies, describing contents, provenance, and accuracy [36]. In many cases, authentication, authorization, and management mechanisms and policies are also required to control who can access services.

- The **hardware infrastructure**, operational support, and policies that allow services to be operated in a suitably convenient, reliable, secure, and performant manner, and that permit users to access services efficiently over local and wide area networks.
- A community of developers, operators, and users who have the technical expertise required to construct, operate, and use services. New approaches to **education and training** may be required to develop this community.

Note that success in each area depends on both technological and sociological issues. Indeed, the nontechnical issue of incentives may be the most important of all. A scientist may work long hours in the pursuit of not only knowledge but also tenure, fame, and/or fortune. The same time spent developing a service may not be so rewarded. We need to change incentives and enable specialization so that being a service developer is as honorable as being an experimentalist or theorist. Intellectual property issues must also be addressed so that people feel comfortable making data available freely. It is perhaps not surprising that astronomy has led the way in putting data online, given that its data has no known commercial value [39].

1.1. Creating, Discovering, and Accessing Services

For SOS to flourish, we need to kick start a virtuous cycle in which the following steps are performed repeatedly by many participants:

- Users discover interesting data and/or software services, and determine that they meet their purposes;
- They compose this service with others to create new capabilities; and
- They publish the resulting services for use by others (perhaps subject to access control).

We can thus catalyze the creation of distributed networks of services, each constructed by a different individual or group, and each providing some original content and/or value-added product.

The U.S. National Cancer Institute takes SOS seriously. Its caBIG project [34] (Figure 1) seeks to enable new approaches to cancer research and care by facilitating the sharing of data and software across the many cancer centers and related institutions. To this end, caBIG leaders have defined and implemented a comprehensive architecture that addresses every aspect of the service lifecycle, from authoring to publication, discovery, composition, and access. The resulting service oriented architecture builds on Web Services standards, vocabularies and ontologies developed within the medical community, and the Globus open source software [10].

Authoring is assisted by a tool called Introduce [17], which allows users to define stateful Globus-based services, specify deployment parameters, and specify access control policies for the new service. A complementary tool, the Remote Application Virtualization Environment (RAVE), builds on Introduce to allow for the wrapping of arbitrary applications as Web Services. Figure 2 shows the steps involved in Introduce-RAVE service creation and deployment:

1. Using the RAVE-enhanced Introduce, the application service is defined in terms of its executable, the form of its input and output messages, its access control policies, and other metadata.
2. The service code is generated and stored in a repository.
3. The service is also registered in a service registry, so that users can discover its existence.
4. When required (e.g., proactively, or in response to a user request), the service implementation is copied to an execution site …
5. … and deployed.

Once a service is deployed, users can then proceed to discover it and access it in the usual way:

6. A user or program can discover the service's existence …
7. … and invoke it via conventional Web Services mechanisms.

Of course, standard vocabularies are not always a prerequisite for automated analysis. To give one example, the GeneWays system mines the raw biological literature to identify experimentally derived relationships and to infer what credence to put in those relationships [33].

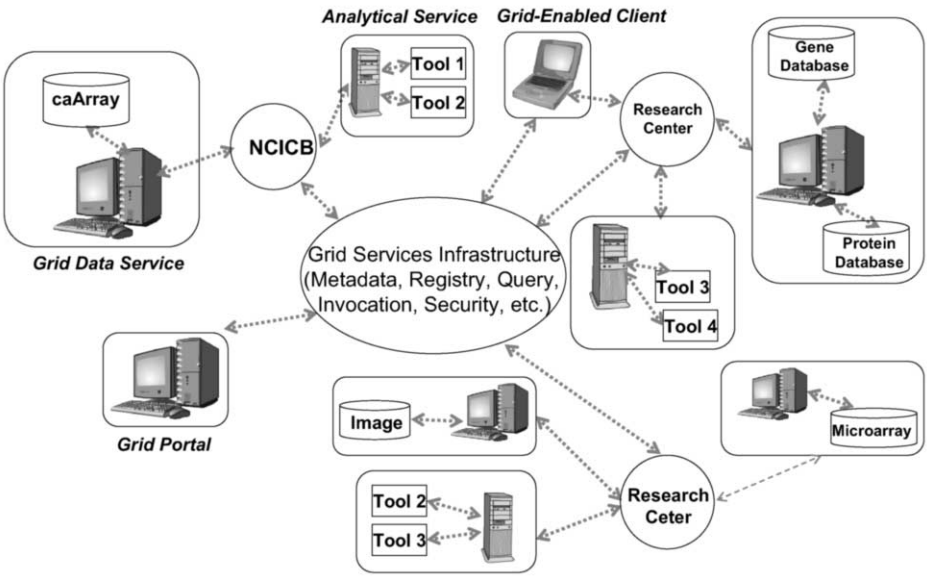


Figure 1: A caBIG deployment, showing data and compute services, portals, NCI infrastructure, and other components

1.2. Hosting and Provisioning

In order for this virtuous cycle to flourish, we must both minimize the costs of not only creating but also operating services and also make it possible to build services that can scale to meet application demands. Thus, we require efficient and convenient service hosting mechanisms. We take such mechanisms for granted when it comes to

Web pages—few people run their own Web server nowadays—but they are still rare for services. These mechanisms should allow for the rapid and convenient deployment of new services, for the dynamic provisioning of services in response to changing demand, for access control, and for accounting and audit.

Service deployment mechanisms need (in one way or another) to acquire required resources at a hosting site, configure those resources appropriately, install and configure service code, and initiate the service. Globus Toolkit support for these functions illustrates some of the different ways in which they can be achieved:

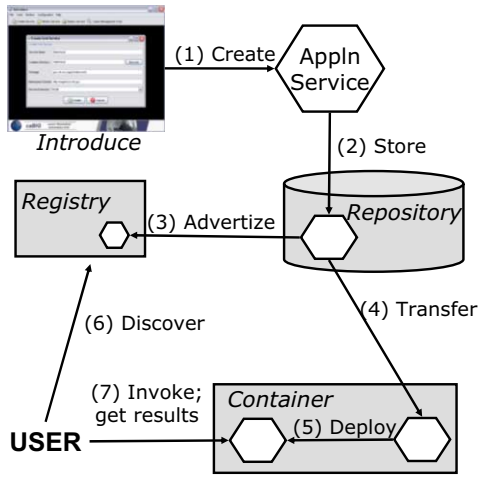


Figure 2: Service creation and deployment steps

- **Dynamic deployment** of Java Web Services into an existing container allows for the rapid and lightweight creation of new services [31]. However, this approach only works for Java Web Services, and the Apache Axis container that Globus uses does not provide for resource management among different services running in the same container.
- The **GRAM service** provides for the dynamic instantiation, and subsequent monitoring and management, of arbitrary executable programs, which may themselves then operate as services [9]. GRAM is widely used for service deployment, for example when "gliding in" Condor, Sun Grid Engine [41], or Falkon agents. However, GRAM only provides limited control over how the computational resource is configured.
- The **virtual workspace service** [19] provides for the dynamic deployment, and subsequent monitoring and management, of arbitrary virtual machine (VM) images. A VM instance provides a high degree of control over execution environment and resource allocations, but is a more heavyweight construct than a process.

Amazon's Extensible Computing Cloud (EC2) is one of several providers operating on-demand computing resources. Like the virtual workspace service, it provides a Web Services interface for virtual machine deployment and management; however, it provides only simple security mechanisms. A workspace service interface to EC2 makes it easy for service creators to deploy services onto EC2 resources.

Once a service is deployed, clients need to be able to monitor and manage its execution. They may also want to negotiate support for dynamic provisioning, i.e., for varying the resources allocated to a service in response to changing load. Services must often deal with data volumes, computational demands, and numbers of users beyond the capacity of a typical PC. Responding to a user request—or to the arrival of new data—can involve large amounts of computation. For example, the Argonne GNARE system searches periodically through DNA and protein databases for new and updated genomes and then computes and publishes derived values [37]. Analysis of a single

bacterial genome of 4,000 sequences by three bioinformatics tools (BLAST, PFAM, and BLOCKS) requires 12,000 steps, each taking on the order of 30 seconds of run time. GNARE is able to perform these tasks in a timely fashion only because it has access to distributed resources provided by two U.S. national-scale infrastructures, TeraGrid [7] and Open Science Grid [38].

Dynamic provisioning becomes increasingly important as data analysis tasks are increasingly automated. For example, it is improbable that even a tiny fraction of the perhaps 500,000 biologists worldwide will decide to access Genbank, GNARE, or any other service at the same time. However, it is quite conceivable that 50,000 "agents" operating on their behalf would do so—and that each such agent would generate thousands of requests.

IBM's Oceano project [4] pioneered important ideas in dynamic provisioning, which is now becoming quasi-mainstream in certain commercial sectors. In my group, we are applying dynamic provisioning to both individual scientific applications and to scientific workloads with time varying resource demands. Falkon [32] monitors application load and then uses GRAM commands to acquire and release resources. By varying resource acquisition and release policies, we can tradeoff responsiveness to user requests and total resource consumption.

Finally, in a networked world, any useful service will become overloaded. Thus, we need to control who uses services and for what purposes. Particularly valuable services may become community resources requiring coordinated management. Grid architectures and software can play an important role [10]. We also need to be concerned with ensuring that SOS realize its promise of being a democratizing force, rather than increasing the gap between the "haves" and "have-nots."

2. Provenance

Progress in science depends on one researcher's ability to build on the results of another. SOS can make it far easier, from a mechanical perspective, for researchers to do just this, by using service invocations to perform data access, comparison, and analysis tasks that might previously have required manual literature searches, analyses, or even experiments. However, the results of these activities are only useful when published if other researchers can determine how much credence to put in the results on which they build, and in turn convince their peers that their results are credible.

One approach to this problem emphasizes reputation as the primary basis for evaluating and enforcing quality [44]. If each published result is associated with an author, then others can judge whether to trust a result based on their prior experience with results published by that author—and the author, being concerned with their reputation, will seek to maximize quality. This process is, in essence, that followed with print publications today, with the rigor of the reviewing process in a particular journal or conference also playing a role.

However, while reputation certainly has a role to play in trust, few researchers will be comfortable trusting a result on that sole basis. They will also want to see details on the method used to obtain a result. Such information can be used to determine whether a result can be trusted, can provide insights into when and where the result can be trusted, and can help guide new research.

Such documentation corresponds, in a broad sense, to the "methods" section in experimental papers, which should in principle provide enough information to allow a

researcher to replicate an experiment. While that principle is perhaps honored more in the breach than in the observance, it is still a fundamental concept for science.

Increased use of computational techniques introduces new challenges to the documentation of experimental procedures (e.g., what version of software was used? what parameters were set?), but also offers the potential for significant improvements in "reproducibility." After all, while it may be impossible to capture the exact actions performed by an experimental scientist, the digital nature of computations means that it can be possible (in principle) to capture the exact sequence of computational steps performed during simulation or analysis.

These observations have motivated growing interest in methods for recording the provenance of computational results. Initial work focused on databases [6, 43], but interest has broadened to encompass arbitrary computations [14, 23]. A series of workshops [24] have led to the formulation of a provenance challenge [25], in which many groups have participated. Approaches explored include the use of functional scripting languages to express application tasks [45], file system instrumentation [27], and the use of a general-purpose provenance store [23].

3. Building Communities

Research occurs within communities, and the formation and operation of communities can be enabled by appropriate technology. Thus, Bill Wulf introduced in 1993 the concept of the collaboratory:

> *a center without walls, in which the nation's researchers can perform their research without regard to geographical location—interacting with colleagues, accessing instrumentation, sharing data and computational resources, and accessing information in digital libraries* [1].

Five years later, Carl Kesselman and I wrote that Grid technologies are concerned with:

> coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. *The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a* virtual organization *(VO).* [13]

These two characterizations capture important aspects of the technology required to enable collaboration within distributed communities, emphasizing in particular the need for shared infrastructure, on-demand access, and mechanisms for controlling community membership and privileges.

While great progress has been made in tools for forming and operating distributed scientific communities, many challenges remain. For example, mechanisms that work effectively for two or ten participants may not scale effectively to one thousand or one

million—not necessarily because implementations cannot handle the number of tasks, but because softer issues such as trust, shared vocabulary, and other implicit knowledge break down as communities extend beyond personal connections.

One approach to solving some scaling problems is to build infrastructures that allow clients to associate arbitrary metadata ("assertions") with data and services. Assuming that we can also determine whether such assertions can be trusted (perhaps on the basis of digital signatures, and/or yet other assertions), consumers can then make their own decisions concerning such properties as quality, provenance, and accuracy. Various popular systems demonstrate the advantages, costs, and pitfalls of different approaches to building such community knowledge bases: for example, the Wikipedia collaborative authoring system, the Flickr and Connotaea collaborative tagging systems [16], and game-based systems for improving tag quality [3].

4. Automating Protocols

Science is not simply a matter of analyzing data or running simulations. Depending on context, it can involve planning and conducting experiments, collecting and analyzing data, deriving models from data, performing many different simulations to explore the implications of models, inferring new hypotheses from data, and planning new experiments. As the complexity of each of these steps increases, each becomes a candidate for automation. Thus, we encounter several related concerns: identifying what to automate, determining how to automate, and documenting automated procedures so that they can understood, validated, and replicated.

In the natural sciences, a protocol is a:

> *predefined written procedural method in the design and implementation of experiments [that] should establish standards that can be adequately assessed by peer review and provide for successful replication of results by others in the field* [2].

These remarks were written in the context of procedures intended to be performed manually, albeit perhaps with the aid of automated equipment. However, they can also apply to procedures applied entirely by computers, in which case we may refer to an *automated protocol*.

Because automated protocols are performed by computers and without human intervention, they can operate far faster than manual protocols. Thus, it becomes increasingly important to document precisely what operations are performed. Arguably, the fact that operations are performed under computer control also makes it more feasible to describe the protocol's operation, although as Muggleton [26] notes, "there is a severe danger that increases in speed and volume of data generation in science could lead to decreases in comprehension of the results."

Two areas in which automation has already had a major impact are data collection and integration [15]. In astronomy, automated sky surveys collect many terabytes of digital data per year, enabling new approaches to astronomy, as discussed above. In biology, the cost of DNA sequencing has reduced from around $10 per base pair in 1990 to less than 1 cent per base pair today. Thus, it becomes possible to perform, for example, "genetic surveys" of many species, and integrate new data from different sources [29, 37].

Sky surveys and genome sequencing both involve a comprehensive survey of an entire object (the sky or genome). In other cases, decision procedures are required to guide data collection, as when searching for transient events in astronomy or when exploring combinatorial spaces, such as the result of one (set of) experiment(s) helps guide the selection of the next. In that case, automated protocols can involve not only data collection and analysis but also the decision procedures used to operate experimental apparatus. In one suggestive project, King et al. [20] describe a "robot scientist" that uses automated mechanisms to identify experiments that can discriminate among competing hypotheses. They report that their best algorithm can outperform humans in terms of number of experiments required to achieve a given accuracy of prediction. Such algorithms may become a standard part of the scientist's repertoire, and future papers may note that "we obtained these results using equipment X controlled by algorithm Y."

Technological improvements continue to reduce the cost and increase the speed of experimental apparatus. For example, microfluidic devices allow for cheaper and more easily automated laboratory experiments, by allowing the delivery of precise and minute quantities of experimental reagents. In an interesting twist, Prakash and Gershenfeld [30] describe how such apparatus can be controlled by embedded digital control, via what they call microfluidic bubble logic. Thus experimental protocols may extend to the configuration of multiple forms of digital and analog devices.

5. Summary

When Licklider expressed his vision of computer-human symbiosis, he was restating, in terms of the technology of his day, and with a particular focus on problem solving, Alfred Whitehead's observation that:

> *Civilization advances by extending the number of important operations which we can perform without thinking about them*. [42]

The computer has greatly expanded the number of operations that we can perform without thinking. However, as we have discussed in this paper, the number of operations that remain susceptible to automation remains large—indeed, is perhaps unbounded.

In seeking further opportunities for optimization of human problem solving, we need to take a system-level [12] or end-to-end view, in which we study and seek opportunities for optimization in every aspect of the problem solving process, not only by the individual researcher or within an individual laboratory, but also within and across communities. For example, we may determine that (as I have argued here) service oriented architectures can be used to distribute and thus accelerate the processes of publishing, discovering, and accessing relevant data and software; that the encoding of provenance information can facilitate the reuse of computational resources; that software support for building communities can promote the collaborative development of knowledge; and that the representation as data objects of the protocols used to perform experiments, analyze data, construct simulations, test simulation codes, and so forth, can raise the level at which the results of thinking ("cognitive artifacts") are reused. Many other opportunities can easily be identified.

In examining these issues, I have focused on the concerns of scientists and science. Scientists are certainly not alone in grappling with these issues. However, science is perhaps unique in the scope and scale of its problems and the subtlety of the questions that the methods discussed here can be used to answer. We may expect that methods developed for science can find application elsewhere, even as scientists look increasingly to computer science and information technology for tools that maximize the time that they spend thinking.

Acknowledgements

I am grateful to Lucio Grandinetti for the opportunity to present some of these ideas at the HPC 2006 conference in Cetraro, Italy, and to many colleagues for discussions on these topics, notably Charlie Catlett, Carl Kesselman, Rick Stevens, and Mike Wilde. This work was supported by the Mathematical, Information, and Computational Sciences Division of the Office of Advanced Scientific Computing Research, U.S. Department of Energy (DE-AC02-06CH11357).

References

1. *National Collaboratories: Applying Information Technology for Scientific Research*. National Research Council, 1993.
2. Protocol, http://en.wikipedia.org/wiki/Protocol_(natural_sciences), 2007.
3. Ahn, L.v. Games with a Purpose. *IEEE Computer* (June). 96-98. 2006.
4. Appleby, K., Fakhouri, S., Fong, L., Goldszmidt, G., Kalantar, M., Krishnakumar, S., Pazel, D., Pershing, J. and Rochwerger, B., Oceano - SLA Based Management of a Computing Utility. in *7th IFIP/IEEE International Symposium on Integrated Network Management*, (2001).
5. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. *Web Services Architecture*. W3C, http://www.w3.org/TR/2003/WD-ws-arch-20030808/, 2003.
6. Buneman, P., Khanna, S. and Tan, W.-C., Why and Where: A Characterization of Data Provenance. in *International Conference on Database Theory*, (2001).
7. Catlett, C. and others, TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications. in *HPC 2006*, (2007).
8. Engelbart, D.C. and English, W.K. A Research Center for Augmenting Human Intellect. in *Proceedings of AFIPS 1968 Fall Joint Computer Conference*, San Francisco, CA, 1968.
9. Feller, M., Foster, I. and Martin, S., GT4 GRAM: A Functionality and Performance Study. in *TeraGrid 07*, (2007).
10. Foster, I. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computational Science and Technology*, *21* (4). 523-530. 2006.
11. Foster, I. Service-Oriented Science. *Science*, *308*. 814-817. 2005.
12. Foster, I. and Kesselman, C. Scaling System-level Science: Scientific Exploration and IT Implications. *IEEE Computer* (November). 32-39. 2006.
13. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, *15* (3). 200-222. 2001.
14. Foster, I., Voeckler, J., Wilde, M. and Zhao, Y., The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration. in *Conference on Innovative Data Systems Research*, (2003).
15. Goble, C., Pettifer, S. and Stevens, R. Knowledge Integration: In silico Experiments in Bioinformatics. in *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 2004.
16. Golder, S. and Huberman, B.A. The Structure of Collaborative Tagging Systems. *Journal of Information Science*, *32* (2). 198-208. 2006.
17. Hastings, S.L., Oster, S., Langella, S., Ervin, D.W., Kurc, T.M. and Saltz, J.H. Introduce: An Open Source Toolkit for Rapid Development of Strongly Typed Grid Services. *Journal of Grid Computing*. 2007.

18. Hey, A.J.G. and Trefethen, A. The Data Deluge: An e-Science Perspective. in Berman, F., Fox, G.C. and Hey, A.J.G. eds. *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons Publisher, Inc., 2003.
19. Keahey, K., Foster, I., Freeman, T. and Zhang, X. Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. *Scientific Programming*, *13* (4). 265-275. 2005.
20. King, R.D., Whelan, K.E., Jones, F.M., Reiser, P.J.K., Bryant, C.H., Muggleton, S., Kell, D.B. and Oliver, S. Functional Genomic Hypothesis Generation and Experimentation by a Robot Scientist. *Nature*, *427*. 247-252. 2004.
21. Licklider, J.C.R. and Taylor, R.W. The Computer as a Communication Device. *Science and Technology* (April). 1968.
22. Licklider, J.R. Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics*, *HFE-1*. 4-11. 1960.
23. Miles, S., Groth, P., Branco, M. and Moreau, L. The requirements of using provenance in e-science experiments. *Journal of Grid Computing*, *5* (1). 1-25. 2005.
24. Moreau, L. and Foster, I. (eds.). *International Provenance and Annotation Workshop (IPAW'06)*. Springer LNCS, 2006.
25. Moreau, L., Ludaescher, B., Altintas, I., Barga, R., Bowers, S., Callahan, S., Chin Jr, G., Clifford, B., Cohen, S., Cohen-Boulakia, S., Davidson, S., Deelman, E., Digiampietri, L., Foster, I., Freire, J., Frew, J., Futrelle, J., Gibson, T., Gil, Y., Goble, C., Golbeck, J., Groth, P., Holland, D., Jiang, S., Kim, J., Koop, D., Krenek, A., McPhillips, T., Mehta, G., Miles, S., Metzger, D., Munroe, S., Myers, J., Plale, B., Podhorszki, N., Ratnakar, V., Santos, E., Scheidegger, C., Schuchardt, K., Seltzer, M. and Simmhan, Y. The First Provenance Challenge. *Concurrency and Computation: Practice and Experience*. 2007.
26. Muggleton, S. Exceeding Human Limits. *Nature*, *440*. 409-410. 2006.
27. Muniswamy-Reddy, K., Holland, D., Braun, U. and Seltzer, M., Provenance-Aware Storage Systems. in *2006 USENIX Annual Technical Conference*, (Boston, MA, 2006).
28. Ochsenbein, F., Williams, R., Davenhall, C., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., A. Szalay, Taylor, M. and Wicenec, A. *VOTable Format Definition Version 1.1*. International Virtual Observatory Alliance, 2004.
29. Overbeek, R., Larsen, N., Pusch, G.D., D'Souza, M., Selkov Jr, E., Kyrpides, N., Fonstein, M., Maltsev, N. and Selkov, E. WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic Acids Research*, *28* (1). 123-125. 2000.
30. Prakash, M. and Gershenfeld, N. Microfluidic Bubble Logic. *Science*, *315* (5813). 832-835. 2007.
31. Qi, L., Jin, H., Foster, I. and Gawor, J. HAND: Highly Available Dynamic Deployment Infrastructure for Globus Toolkit 4. 2006.
32. Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I. and Wilde, M. Falkon: a Fast and Light-weight tasK executiON framework for Grid Environments, 2007.
33. Rzhetsky, A., Iossifov, I., Koike, T., Krauthammer, M., Kra, P., Morris, M., Yu, H., Duboue, P.A., Weng, W., Wilbur, W.J., Hatzivassiloglou, V. and Friedman, C. GeneWays: a system for extracting, analyzing, visualizing, and integrating molecular pathway data. *J. Biomed. Inform.*, *37*. 43-53. 2004.
34. Saltz, J.H., Oster, S., Hastings, S.L., Langella, S., Sanchez, W., Kher, M. and Covitz, P.A. caGrid: design and implementation of the core architecture of the cancer biomedical informatics grid. *Bioinformatics*, *22* (15). 1910-1916. 2006.
35. Schena, M., Shalon, D., Davis, R. and Brown, P. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, *270*. 467-470. 1995.
36. Stein, L. Creating a Bioinformatics Nation. *Nature*, *317*. 119-120. 2002.
37. Sulakhe, D., Rodriguez, A., D'Souza, M., Wilde, M., Nefedova, V., Foster, I. and Maltsev, N. GNARE: An Environment for Grid-Based High-Throughput Genome Analysis. *Journal of Clinical Monitoring and Computing*. 2005.
38. Sulakhe, D., Rodriguez, A., Wilde, M., Foster, I. and Maltsev, N., Using Multiple Grid Resources for Bioinformatics Applications in GADU. in *IEEE/ACM International Symposium on Cluster Computing and Grid*, (2006).
39. Szalay, A. and Gray, J. The World-Wide Telescope. *Science*, *293*. 2037-2040. 2001.
40. Tsvetanov, Z.I., Golimowski, D.A., Zheng, W., Geballe, T.R., Leggett, S.K., Ford, H.C., Davidsen, A.F., Uomoto, A., Fan, X., Knapp, G.R., Strauss, M.A., Brinkmann, J., Lamb, D.Q., Newberg, H.J., Rechenmacher, R., Schneider, D.P., York, D.G., Lupton, R.H., Pier, J.R., Annis, J., Csabai, I., Hindsley, R.B., Ivesic, Z., Munn, J.A., Thakar, A.R. and Waddell, P. The Discovery of a Second Field Methane Brown Dwarf from Sloan Digital Sky Survey Commissioning Data. *Astrophysical Journal*, *531*. L61. 2000.
41. Walker, E., Gardner, J.P., Litvin, V. and Turner, E.L. Creating personal adaptive clusters for managing scientific jobs in a distributed computing environment *Challenges of Large Applications in Distributed Environments*, IEEE, 2006.

42. Whitehead, A.N. *Introduction to Mathematics*, 1911.
43. Woodruff, A. and Stonebraker, M., Supporting Fine-Grained Data Lineage in a Database Visualization Environment. in *13th International Conference on Data Engineering*, (1997), 91-102.
44. Zacharia, G. and Maes, P. Trust Management Through Reputation Mechanisms. *Applied Artificial Intelligence*, *14*. 881-907. 2000.
45. Zhao, Y., Dobson, J., Foster, I., Moreau, L. and Wilde, M. A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data. *SIGMOD Record 34* (3). 37-43 2005.

This page intentionally left blank

Chapter 1

Systems and Solutions for Advanced Distributed Computing and for High Performance Computing and Networking

This page intentionally left blank

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Exploiting Mixed Precision Floating Point Hardware in Scientific Computations

Alfredo BUTTARI [a] Jack DONGARRA [a,b] Jakub KURZAK [a] Julie LANGOU [a]
Julien LANGOU [c] Piotr LUSZCZEK [a] and Stanimire TOMOV [a]
[a] *Department of Computer Science, University of Tennessee Knoxville*
[b] *Oak Ridge National Laboratory*
[c] *University of Colorado at Denver and Health Sciences Center*

Abstract. By using a combination of 32-bit and 64-bit floating point arithmetic, the performance of many dense and sparse linear algebra algorithms can be significantly enhanced while maintaining the 64-bit accuracy of the resulting solution. The approach presented here can apply not only to conventional processors but also to exotic technologies such as Field Programmable Gate Arrays (FPGA), Graphical Processing Units (GPU), and the Cell BE processor. Results on modern processor architectures and the Cell BE are presented.

Keywords. Iterative refinement, factorization, Krylov methods

Introduction

In numerical computing, there is a fundamental performance advantage in using the single precision, floating point data format over the double precision one. Due to more compact representation, twice the number of single precision data elements can be stored at each level of the memory hierarchy including the register file, the set of caches, and the main memory. By the same token, handling single precision values consumes less bandwidth between different memory levels and decreases the number of cache and TLB misses. However, the data movement aspect affects mostly memory-intensive, bandwidth-bound applications, historically have not drawn much attention to mixed precision algorithms.

In the past, the situation looked differently for computationally intensive workloads, where the load was on the floating point processing units rather than the memory subsystem, and so the single precision data motion advantages were for the most part irrelevant. With the focus on double precision in the scientific computing, double precision execution units were fully pipelined and capable of completing at least one operation per clock cycle. In fact, in many high performance processor designs single precision units were eliminated in favor of emulating single precision operations using double precision circuitry. At the same time, a high degree of instruction level parallelism was being achieved by introducing more functional units and relatively complex speculation mechanisms, which did not necessarily guarantee full utilization of the hardware resources.

Table 1. Floating point performance characteristics of *individual cores* of modern, multi-core processor architectures.

Architecture	Clock [GHz]	DP Peak [Gflop/s]	SP Peak [Gflop/s]
AMD Opteron 246	2.0	4	8
IBM PowerPC 970	2.5	10	20
Intel Xeon 5100	3.0	12	24
STI Cell BE	3.2	1.8[1]	25.6

That situation began to change with the widespread adoption of short vector, Single Instruction Multiple Data (SIMD) processor extensions, which started appearing in the mid 90s. An example of such extensions are the Intel MultiMedia eXtensions (MMX) that were mostly meant to improve processor performance in Digital Signal Processing (DSP) applications, graphics and computer games. Short vector, SIMD instructions are a relatively cheap way of exploiting data level parallelism by applying the same operation to a vector of elements at once. It eliminates the hardware design complexity associated with the bookkeeping involved in speculative execution. It also gives better guarantees for practically achievable performance than does runtime speculation, provided that enough data parallelism exists in the computation. Most importantly, short vector, SIMD processing provides the opportunity to benefit from replacing the double precision arithmetic with the single precision one. Since the goal is to process the entire vector in a single operation, the computation throughput doubles while the data storage space halves.

Most processor architectures available today have been augmented, at some point, in their design evolution with short vector, SIMD extensions. Examples include Streaming SIMD Extensions (SSE) for the AMD and the Intel line of processors; PowerPC's Velocity Engine, AltiVec, and VMX; SPARC's Visual Instruction Set (VIS); Alpha's Motion Video Instructions (MVI); PA-RISC's Multimedia Acceleration eXtensions (MAX); MIPS-3D Application Specific Extensions (ASP) and Digital Media Extensions (MDMX) and ARM's NEON feature. The different architectures exhibit large differences in their capabilities. The vector size is either 64 bits or, more commonly, 128 bits. The register file size ranges from just a few to as many as 256 registers. Some extensions only support integer types while others operate on single precision, floating point numbers, and yet others process double precision values.

Today, the Synergistic Processing Element (SPE) of the CELL processor can probably be considered the state of the art in short vector, SIMD processing. Possessing 128-byte long registers and a fully pipelined fused, multiply-add instruction, it is capable of completing as many as eight single precision, floating point operations each clock cycle. When combined with the size of the register file of 128 registers, it is capable of delivering close to peak performance on many common computationally intensive workloads.

Table 1 shows the difference in peak performance between single precision (SP) and double precision (DP) of four modern processor architectures. Following the recent trend in chip design, all of the presented processors are multi-core architectures. However, to avoid introducing the complexity of thread-level parallelization to the discussion, we will mainly look at the performance of individual cores throughout the chapter. The goal here is to focus on instruction-level parallelism exploited by short vector SIMD'zation.

[1] The DP unit is not fully pipelined, and has a 7 cycle latency.

Although short vector, SIMD processors have been around for over a decade, the concept of using those extensions to utilize the advantages of single precision performance in scientific computing did not come to fruition until recently, due to the fact that most scientific computing problems require double precision accuracy. It turns out, however, that for many problems in numerical computing, it is possible to exploit the speed of single precision operations and resort to double precision calculations at few stages of the algorithm to achieve full double precision accuracy of the result. The techniques described here are fairly general and can be applied to a wide range of problems in linear algebra, such as solving linear systems of equations, least square problems, singular value and eigenvalue problems. Here we are going to focus on solving linear systems of equations, both dense and sparse, non-symmetric and symmetric, using direct methods, as well as iterative, Krylov subspace methods.

In this paper, we are going to focus on solving linear systems of equations, non-symmetric and symmetric, dense (Section 1) and sparse, using direct methods (Section 2), as well as iterative, Krylov subspace methods (Section 3).

1. Direct Methods for Solving Dense Systems

1.1. Algorithm

Iterative refinement is a well known method for improving the solution of a linear system of equations of the form $Ax = b$ [1]. The standard approach to the solution of dense linear systems is to use the LU factorization by means of Gaussian elimination. First, the coefficient matrix A is factorized into the product of a lower triangular matrix L and an upper triangular matrix U using LU decomposition. Commonly, partial row pivoting is used to improve numerical stability resulting in the factorization $PA = LU$, where P is the row permutation matrix. The solution for the system is obtained by first solving $Ly = Pb$ (*forward substitution*) and then solving $Ux = y$ (*back substitution*). Due to the round-off error, the computed solution x carries a numerical error magnified by the condition number of the coefficient matrix A. In order to improve the computed solution, an iterative refinement process is applied, which produces a correction to the computed solution at each iteration, which then yields the basic iterative refinement algorithm (Algorithm 1). As Demmel points out [2], the non-linearity of the round-off error makes the iterative refinement process equivalent to the Newton's method applied to the function $f(x) = b - Ax$. Provided that the system is not too ill-conditioned, the algorithm produces a solution correct to the working precision. Iterative refinement is a fairly well understood concept and was analyzed by Wilkinson [3], Moler [4] and Stewart [1].

The algorithm can be modified to use a mixed precision approach. The factorization $PA = LU$ and the solution of the triangular systems $Ly = Pb$ and $Ux = y$ are computed using single precision arithmetic. The residual calculation and the update of the solution are computed using double precision arithmetic and the original double precision coefficients. The most computationally expensive operations, including the factorization of the coefficient matrix A and the forward and backward substitution, are performed using single precision arithmetic and take advantage of its higher speed. The only operations that must be executed in double precision are the residual calculation and the update of the solution. It can be observed, that all operations of $O(n^3)$ computational complex-

Algorithm 1 The iterative refinement method for the solution of linear systems

1: $x_0 \leftarrow A^{-1}b$
2: $k = 1$
3: **repeat**
4: $\quad r_k \leftarrow b - Ax_{k-1}$
5: $\quad z_k \leftarrow A^{-1}r_k$
6: $\quad x_k \leftarrow x_{k-1} + z_k$
7: $\quad k \leftarrow k + 1$
8: **until** convergence

ity are handled in single precision, and all operations performed in double precision are of at most $O(n^2)$ complexity. The coefficient matrix A is converted to single precision for the LU factorization and the resulting factors are also stored in single precision. At the same time, the original matrix in double precision must be preserved for the residual calculation. The mixed precision, iterative refinement algorithm is outlined in Algorithm 2; the (32) subscript means that the data is stored in 32-bit format (i.e., single precision) and the absence of any subscript means that the data is stored in 64-bit format (i.e., double precision). Implementation of the algorithm is provided in the LAPACK package by the routine DSGESV.

Algorithm 2 Solution of a linear system of equations using mixed precision, iterative refinement. (SGETRF and SGETRS are names of LAPACK routines).

$A_{(32)}, b_{(32)} \leftarrow A, b$
$L_{(32)}, U_{(32)}, P_{(32)} \leftarrow \text{SGETRF}(A_{(32)})$
$x^{(1)}_{(32)} \leftarrow \text{SGETRS}(L_{(32)}, U_{(32)}, P_{(32)}, b_{(32)})$
$x^{(1)} \leftarrow x^{(1)}_{(32)}$
$i \leftarrow 0$
repeat
$\quad i \leftarrow i + 1$
$\quad r^{(i)} \leftarrow b - Ax^{(i)}$
$\quad r^{(i)}_{(32)} \leftarrow r^{(i)}$
$\quad z^{(i)}_{(32)} \leftarrow \text{SGETRS}(L_{(32)}, U_{(32)}, P_{(32)}, r^{(i)}_{(32)})$
$\quad z^{(i)} \leftarrow z^{(i)}_{(32)}$
$\quad x^{(i+1)} \leftarrow x^{(i)} + z^{(i)}$
until $x^{(i)}$ is *accurate enough*

Higham [5] gives error bounds for the single and double precision, iterative refinement algorithm when the entire algorithm is implemented with the same precision (single or double, respectively). He also gives error bounds in single precision arithmetic, with refinement performed in double precision arithmetic [5]. The error analysis in double precision, for our mixed precision algorithm (Algorithm 2), is given by Langou et al. [6].

The same technique can be applied to the case of symmetric, positive definite problems. Here, Cholesky factorization (LAPACK's SPOTRF routine) can be used in place of LU factorization (SGETRF), and a symmetric back solve routine (SPOTRS) can be used in place of the routine for the general (non-symmetric) case (SGETRS). Also, the matrix-

Table 2. Hardware and software details of the systems used for performance experiments.

Architecture	**Clock [GHz]**	Memory **[MB]**	BLAS	Compiler
AMD Opteron 246	2.0	2048	Goto-1.13	Intel-9.1
IBM PowerPC 970	2.5	2048	Goto-1.13	IBM-8.1
Intel Xeon 5100	3.0	4096	Goto-1.13	Intel-9.1
STI Cell BE	3.2	512	–	Cell SDK-1.1

vector product Ax can be implemented by the BLAS' DSYMV routine, or DSYMM for multiple right hand sides, instead of the DGEMV and DGEMM routines for the non-symmetric case. The mixed precision algorithm for the symmetric, positive definite case is presented by Algorithm 2. Implementation of the algorithm is provided in the LAPACK package by the routine DSPOSV.

Algorithm 3 Solution of a symmetric positive definite system of linear equations using mixed precision, iterative refinement. (SPOTRF and SPOTRS are names of LAPACK routines).

$A_{(32)}, b_{(32)} \leftarrow A, b$
$L_{(32)}, L^T_{(32)} \leftarrow$SPOTRF$(A_{(32)})$
$x^{(1)}_{(32)} \leftarrow$SPOTRS$(L_{(32)}, L^T_{(32)}, b_{(32)})$
$x^{(1)} \leftarrow x^{(1)}_{(32)}$
$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  $r^{(i)} \leftarrow b - Ax^{(i)}$
  $r^{(i)}_{(32)} \leftarrow r^{(i)}$
  $z^{(i)}_{(32)} \leftarrow$SPOTRS$(L_{(32)}, L^T_{(32)}, r^{(i)}_{(32)})$
  $z^{(i)} \leftarrow z^{(i)}_{(32)}$
  $x^{(i+1)} \leftarrow x^{(i)} + z^{(i)}$
until $x^{(i)}$ is *accurate enough*

1.2. Experimental Results and Discussion

To collect performance results for the Xeon, Opteron and PowePC architectures, the LAPACK iterative refinement routines DSGESV and DSPOSV were used, for the non-symmetric and symmetric cases, respectively. The routines implement classic, blocked versions of the matrix factorizations and rely on the layer of Basic Linear Algebra Subroutines (BLAS) for architecture specific optimizations to deliver performance close to the peak. As mentioned before, in order to simplify the discussion and leave out the aspect of parallelization, we have decided to look at the performance of individual cores on the multi-core architectures.

Figures 1-8 show the performance of the single-core serial implementations of Algorithm 2 and Algorithm 3 on the architectures in Table 2.

These figures show that the mixed precision, iterative refinement method can run very close to the speed of the full single precision solver while delivering the same ac-

curacy as the full double precision one. On the AMD Opteron, Intel Woodcrest and IBM PowerPC architectures (see Figures 1- 6), the mixed precision, iterative solver can provide a speedup of up to 1.8 for the unsymmetric solver and 1.5 for the symmetric one, if the problem size is big enough. For small problem sizes, in fact, the cost of even a few iterative refinement iterations is high compared to the cost of the factorization and thus, the mixed precision, iterative solver is less efficient than the full double precision one.

For the Cell processor (see Figures 7 and 8), parallel implementations of Algorithms 2 and 3 have been produced in order to exploit the full computational power of the processor. Due to the large difference between the single precision and double precision floating point units (see Table 1), the mixed precision solver performs up to 7× and 11× faster than the double precision peak in the unsymmetric and symmetric, positive definite cases respectively. Implementation details for this case can be found in [7, 8].

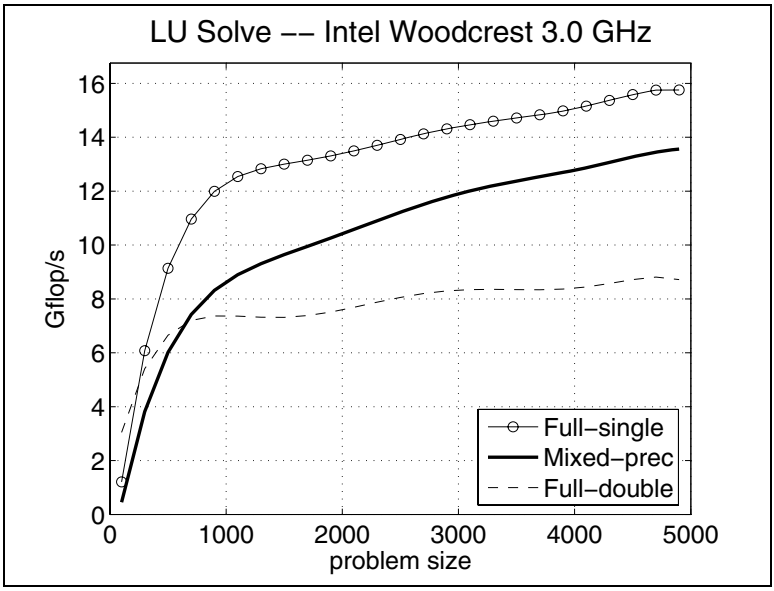


Figure 1. Performance of mixed precision, iterative refinement for unsymmetric problems on Intel Woodcrest.

2. Direct Methods for Solving Sparse Systems

2.1. Algorithm

The mixed precision, iterative refinement methods apply to sparse operations as well as to dense operations. In fact, even for sparse computations, single precision operations are performed at a higher rate than double precision ones. The reason for this difference is different than in the dense case. As already pointed out, in the dense case the difference is due to the fact that vector units in the processors can exploit a higher level of parallelism in the single precision computations than in the double precision ones.

Sparse computations are very difficult to vectorize due to their nature (mostly because of the very irregular memory access patterns and because of the heavy use of in-

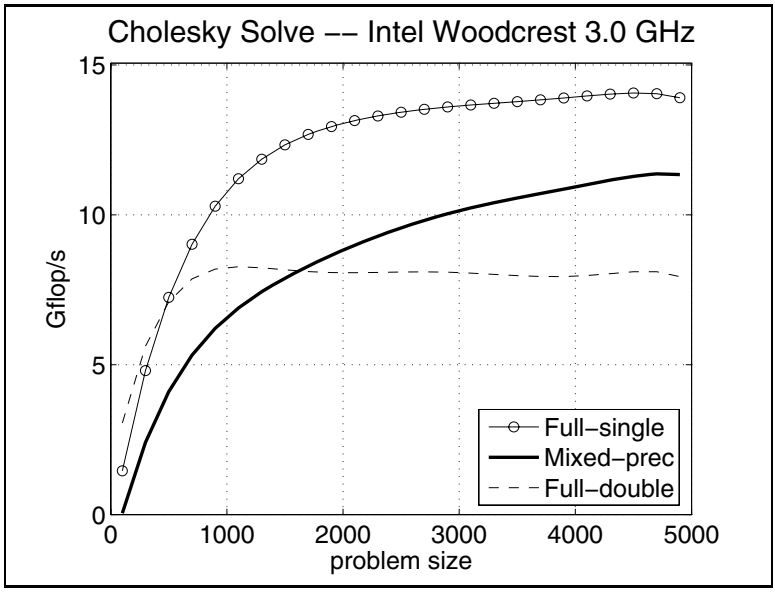


Figure 2. Performance of mixed precision, iterative refinement for symmetric, positive definite problems on Intel Woodcrest.

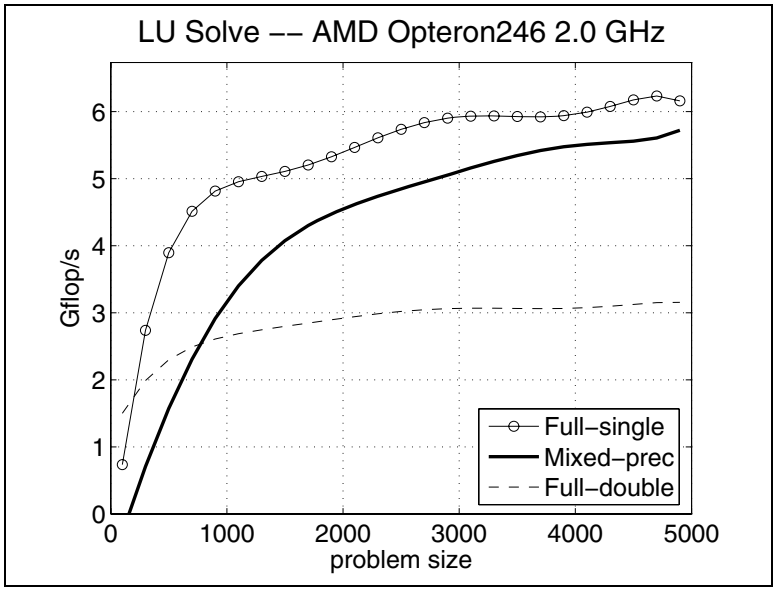


Figure 3. Performance of mixed precision, iterative refinement for unsymmetric problems on AMD Opteron246.

direct addressing). Even in the case where they can be vectorized, this optimization does not have a significant effect on performance because sparse operations are inherently memory bound, which means that the number crunching phase is much cheaper than the cpu-memory communication phase. Despite all this, single precision computations can

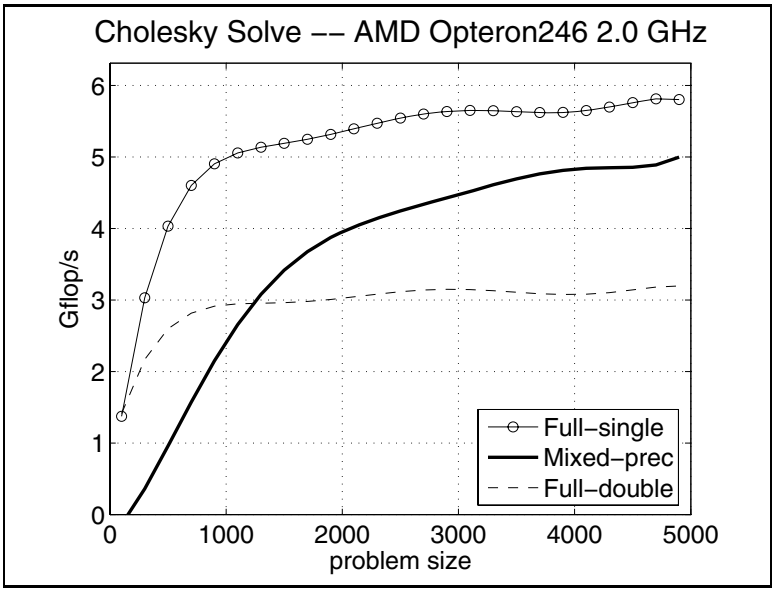


Figure 4. Performance of mixed precision, iterative refinement for symmetric, positive definite problems on AMD Opteron246.

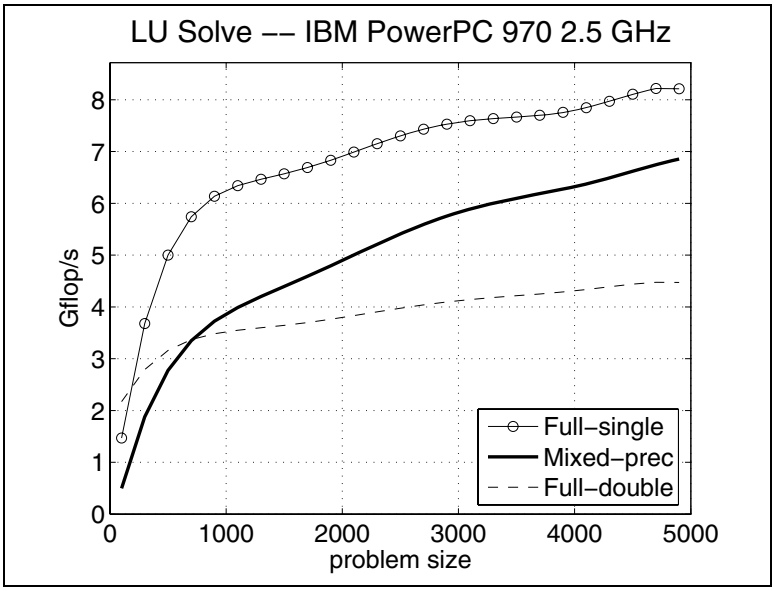


Figure 5. Performance of mixed precision, iterative refinement for unsymmetric problems on IBM PowerPC 970.

be performed at a speed that is up to $2\times$ as fast as in double precision, since the amount of data that is moved through the memory bus is halved. The mixed precision, iterative refinement technique is thus applicable to the solution of sparse linear systems, which is commonly achieved with either direct of iterative methods.

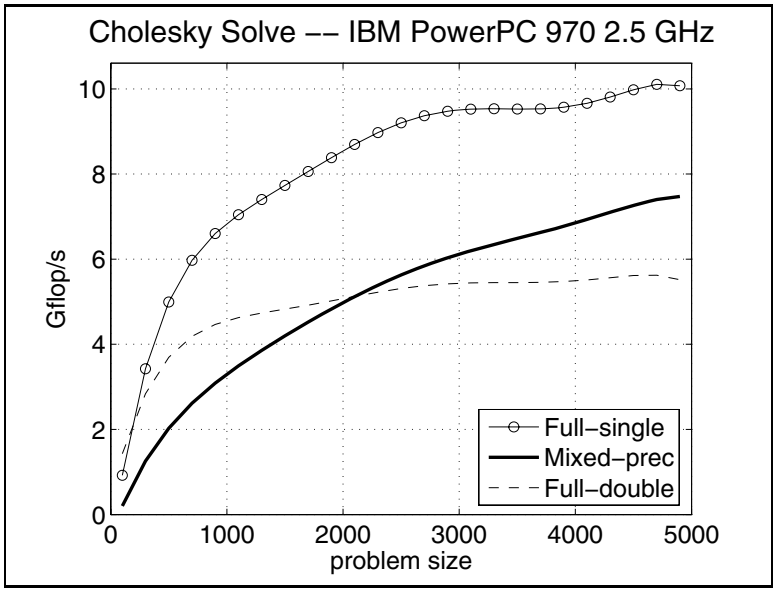


Figure 6. Performance of mixed precision, iterative refinement for symmetric, positive definite problems on IBM PowerPC 970.

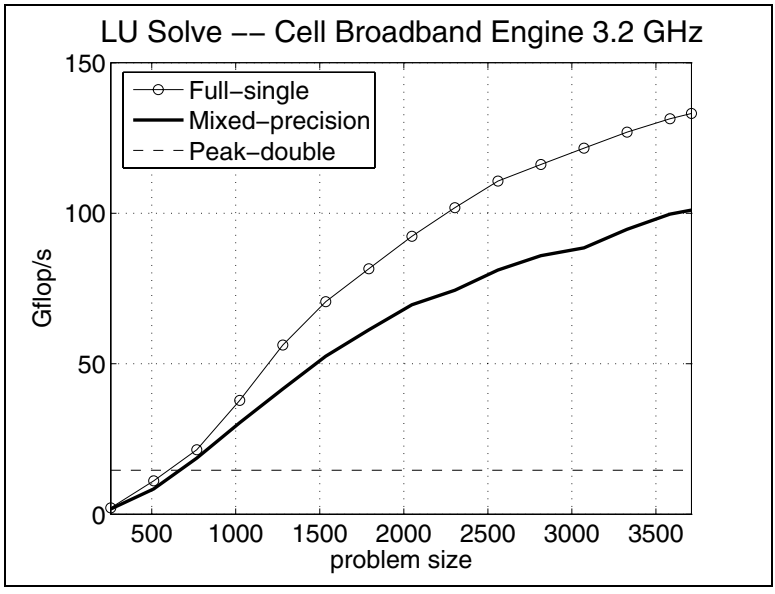


Figure 7. Performance of mixed precision, iterative refinement for unsymmetric problems on CELL Broadband Engine.

Most sparse direct methods for solving linear systems of equations are variants of either multifrontal [9] or supernodal [10] factorization approaches. Here, we focus only on multifrontal methods. For results on supernodal solvers see [11]. There are a number of freely available packages that implement multifrontal methods. We have chosen for

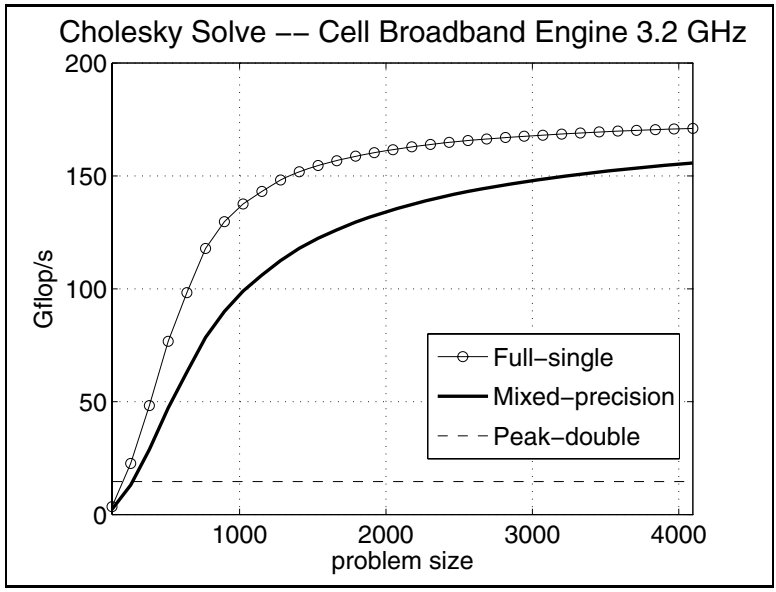


Figure 8. Performance of mixed precision, iterative refinement for symmetric, positive definite problems on CELL Broadband Engine.

our tests the software package called MUMPS [12–14]. The main reason for selecting this software is that it is implemented in both single and double precision, which is not the case for other freely available multifrontal solvers such as UMFPACK [15–17].

Using the MUMPS package for solving systems of linear equations can be described in three distinct steps:

1. System Analysis: in this phase the system sparsity structure is analyzed in order to estimate the element fill-in, which provides an estimate of the memory that will be allocated in the following steps. Also, pivoting is performed based on the structure of $A + A^T$, ignoring numerical values. Only integer operations are performed at this step.
2. Matrix Factorization: in this phase the $PA = LU$ factorization is performed. This is the computationally most expensive step of the system solution.
3. System Solution: the system is solved in two steps: $Ly = Pb$ and $Ux = y$.

Once steps 1 and 2 are performed, each iteration of the refinement loop needs only to perform the system solution (i.e., step 3). The cost of the iterative refinement steps is lower than the advantage obtained by performing the system solution in single precision if the number of iterations is limited. The implementation of a mixed precision, iterative refinement method with the MUMPS package can thus be summarized as in algorithm 4.

At the end of each line of the algorithm, we indicate the precision used to perform this operation as either ϵ_s, for single precision computation, or ϵ_d, for double precision computation. Based on backward stability analysis, the solution x can be considered as accurate as the double precision one when

$$\|b - Ax\|_2 \leq \|x\|_2 \cdot \|A\|_{fro} \cdot \epsilon \cdot \sqrt{n}$$

Algorithm 4 Mixed precision, Iterative Refinement with the MUMPS package

1: system analysis
2: $LU \leftarrow PA$ (ϵ_s)
3: solve $Ly = Pb$ (ϵ_s)
4: solve $Ux_0 = y$ (ϵ_s)
$k \leftarrow 1$
5: **until** convergence **do**:
6: $r_k \leftarrow b - Ax_{k-1}$ (ϵ_d)
7: solve $Ly = Pr_k$ (ϵ_s)
8: solve $Uz_k = y$ (ϵ_s)
9: $x_k \leftarrow x_{k-1} + z_k$ (ϵ_d)
$k \leftarrow k + 1$
10: **done**

where $\| \cdot \|_{fro}$ is the Frobenius norm. The iterative method is stopped when the double precision accuracy is achieved or a maximum number of iterations is reached.

2.2. Experimental Results and Discussion

The method in Algorithm 4 can offer significant improvements for the solution of a sparse linear system in many cases if:

1. the number of iterations is not too high.
2. the cost of each iteration is small as compared to the cost of the system factorization. If the cost of each iteration is too high, then a low number of iterations will result in a performance loss with respect to the full double precision solver. In the sparse case, for a fixed matrix size, both the cost of the system factorization and the cost of the iterative refinement step may substantially vary depending on the number of nonzeroes and the matrix sparsity structure.

The efficiency of the mixed precision, iterative refinement approach on sparse direct solvers is shown in Figures 9, 10 and 11. These figures report the performance ratio between the full single precision and full double precision solvers (light colored bars) and the mixed precision and full-double precision solvers (dark colored bars) for six matrices from real world applications. The number on top of each bar shows how many iterations are performed by the mixed precision, iterative method to achieve double precision accuracy.

The data in Figures 9, 10 and 11 have been measured using the architectures listed in Table 1 (except for the Cell processor) on a number of matrices from real world applications. These matrices are reported in Table 3 and are grouped into symmetric and unsymmetric ones because the MUMPS package uses different numerical methods for these two classes of matrices.

3. Iterative Methods for Solving Sparse Systems

Direct sparse methods suffer from fill-ins and, consequently, high memory requirements as well as extended execution time. There are various reordering techniques designed to

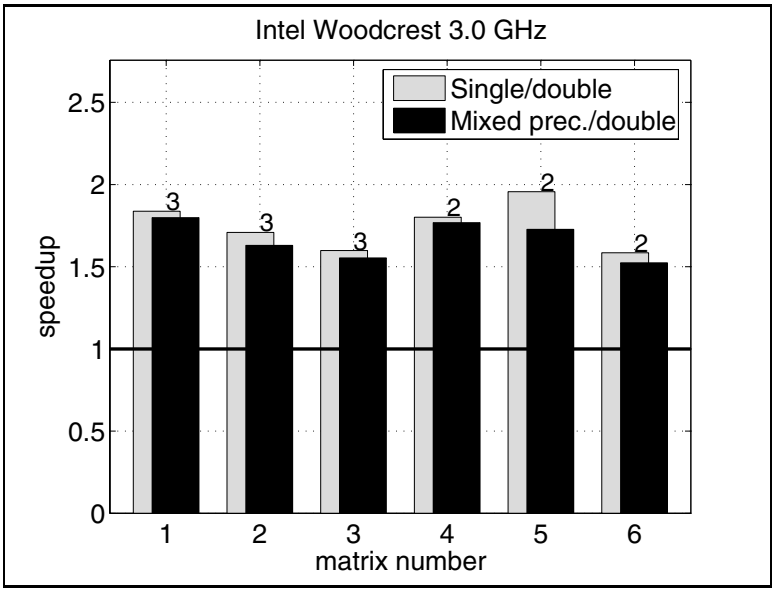


Figure 9. Mixed precision, iterative refinement with the MUMPS direct solver on an Intel Woodcrest 3.0 GHz system.

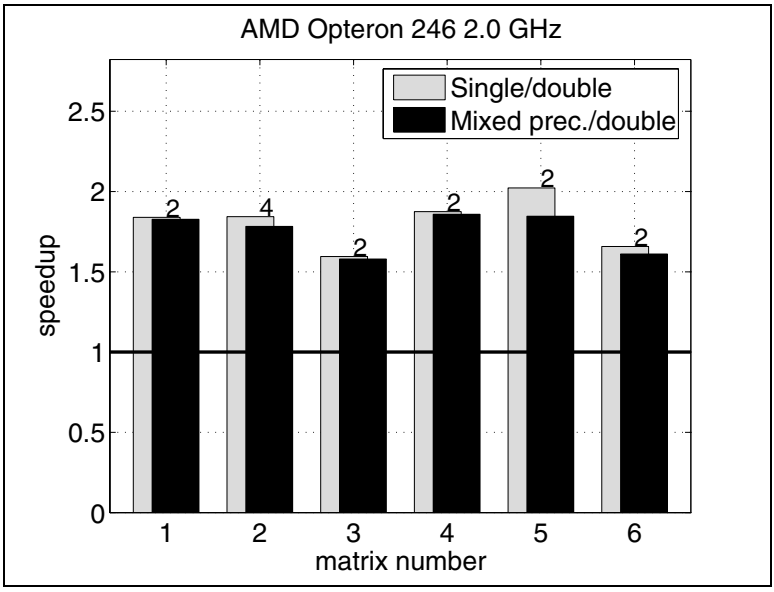


Figure 10. Mixed precision, iterative refinement with the MUMPS direct solver on an AMD Opteron 246 2.0 GHz system.

minimize the amount of fill-ins. Nevertheless, for problems of increasing size, there is a point where they become prohibitively high and direct sparse methods are no longer feasible. Iterative methods are a remedy, since only a few working vectors and the primary data are required [18, 19].

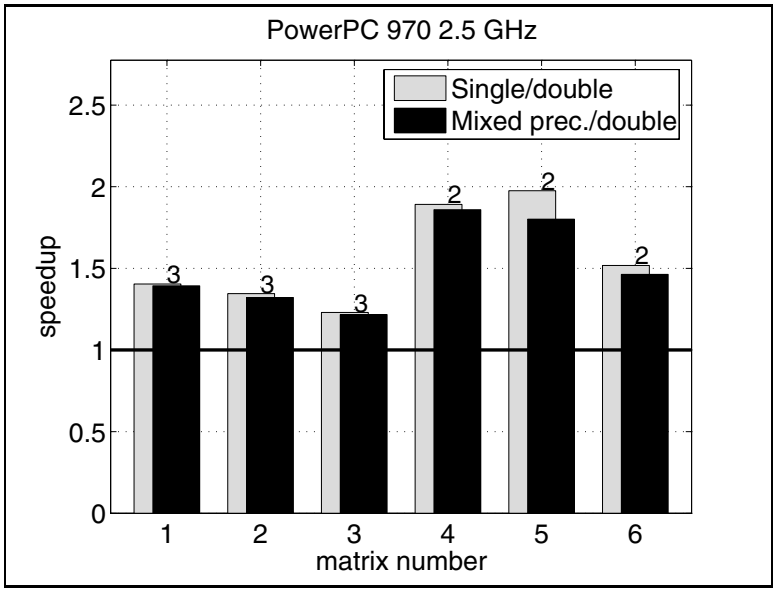


Figure 11. Mixed precision, iterative refinement with the MUMPS direct solver on an IBM PowerPC 970 2.5 GHz system.

Table 3. Test matrices for sparse mixed precision, iterative refinement solution methods.

num.	Matrix	Size	Nonzeroes	symm.	pos. def.	Cond. Numb.
1	SiO	33401	1317655	yes	no	$O(10^3)$
2	Lin	25600	1766400	yes	no	$O(10^5)$
3	c-71	76638	859554	yes	no	$O(10)$
4	cage-11	39082	559722	no	no	$O(1)$
5	raefsky3	21200	1488768	no	no	$O(10)$
6	poisson3Db	85623	2374949	no	no	$O(10^3)$

As an example, let us first consider the iterative refinement itself, described in Algorithm 1 as

$$x_{i+1} = x_i + M(b - Ax_i), \tag{1}$$

where M is $(LU)^{-1}P$. Iterative methods of this form (i.e. where M does not depend on the iteration number i) are also known as *stationary*. Matrix M can be as simple as a scalar value (the method then becomes a modified Richardson iteration) or as complex as $(LU)^{-1}P$. In either case, M is called a *preconditioner*. It should approximate A^{-1}, and the quality of the approximation determines the convergence properties of (1). In general, a preconditioner is intended to improve the robustness and the efficiency of the iterative algorithms. Note that (1) can also be interpreted as a Richardson iteration in solving $MAx = Mb$ (called *left* preconditioning). Another possibility, which we will use in the mixed precision, iterative methods to be described later, is to have *right* preconditioning, where the original problem $Ax = b$ is transformed into a problem of solving

$$AMu = b, \quad x = Mu$$

iteratively. Related to the overall efficiency, M needs to be easy to compute, apply, and store. Note that these requirements were addressed in the mixed precision methods above by replacing M (coming from LU factorization of A followed by matrix inversion), with its single precision representation so that arithmetic operations can be performed more efficiently on it. Here, we go two steps further: we consider replacing not only M by an inner loop of incomplete iterative solver performed in single precision arithmetic [20] , but also the outer loop by more sophisticated iterative methods (e.g., Krylov type).

3.1. Mixed Precision, Inner-Outer Iterative Solvers

Note that replacing M by an iterative method leads to *nesting* of two iterative methods. Variations of this type of nesting, also known in the literature as an *inner-outer* iteration, have been studied, both theoretically and computationally [21–27]. The general appeal of these methods is that computational speedup is possible when the inner solver uses an approximation of the original matrix A that is also faster to apply (e.g., in our case, using single precision arithmetic). Moreover, even if no faster matrix-vector product is available, speedup can often be observed due to improved convergence (e.g., see [23], where Simoncini and Szyld explain the possible benefits of FGMRES-GMRES over restarted GMRES).

To illustrate the above concepts, we demonstrate the ideas with a mixed precision, inner-outer iterative solver that is based on the restarted Generalized Minimal RESidual (GMRES) method. Namely, consider Algorithm 5, where for the outer loop we take the flexible GMRES (FGMRES [19, 22]) and for the inner loop the GMRES in single precision arithmetic (denoted by GMRES$_{SP}$). FGMRES, a minor modification to the standard GMRES, is meant to accommodate non-constant preconditioners. Note that in our case, this non-constant preconditioner is GMRES$_{SP}$. The resulting method is denoted by FGMRES(m_{out})-GMRES$_{SP}$(m_{in}) where m_{in} is the restart for the inner loop and m_{out} for the outer FGMRES.

The potential benefits of FGMRES compared to GMRES are becoming better understood [23]. Numerical experiments confirm cases of improvements in speed, robustness, and sometimes memory requirements for these methods. For example, we show a maximum speedup of close to 15 on the selected test problems. The memory requirements for the method are the matrix A in CRS format, the nonzero matrix coefficients in single precision, 2 m_{out} number of vectors in double precision, and m_{in} number of vectors in single precision.

The Generalized Conjugate Residuals (GCR) method [26, 28] is comparable to the FGMRES and can replace it successfully as the outer iterative solver.

3.2. Numerical Performance

Similar to the case of sparse direct solvers, we demonstrate the numerical performance of Algorithm 5 on the architectures from Table 1 and on the matrices from Table 3.

Figure 12 shows the performance ratio of the mixed precision, inner-outer FGMRES-GMRES$_{SP}$ *vs.* the full, double precision, inner-outer FGMRES-GMRES$_{DP}$, i.e., here we compare two inner-outer algorithms that do the same, with the only difference being that their inner loop's incomplete solvers are performed in correspondingly single and double precision arithmetic.

Algorithm 5 Mixed precision, inner-outer FGMRES(m_{out})-GMRES$_{SP}$(m_{in})

for $i = 0, 1, \ldots$ **do**
  $r = b - Ax_i$
  $\beta = h_{1,0} = ||r||_2$
  check convergence and exit if done
  for $k = 1, \ldots, m_{out}$ **do**
    $v_k = r \ / \ h_{k,k-1}$
    One cycle of GMRES$_{SP}$(m_{in}) in solving $Az_k = v_k$, initial guess $z_k = 0$
    $r = A\ z_k$
    for j=1,...,k **do**
      $h_{j,k} = r^T v_j$
      $r = r - h_{j,k}\ v_j$
    end for
    $h_{k+1,k} = ||r||_2$
    if $h_{k+1,k}$ is small enough **then break**
  end for
  // Define $Z_k = z_1, \ldots, z_k,\ H_k = \{h_{i,j}\}_{1 \le i \le k+1, 1 \le j \le k}$
  Find $W_k = w_1, \ldots, w_k{}^T$ that minimizes $||b - A(x_i + Z_k\ W_k)||_2$
  // note: or equivalently, find W_k that minimizes $||\beta e_1 - H_k\ W_k||_2$
  $x_{i+1} = x_i + Z_k\ W_k$
end for

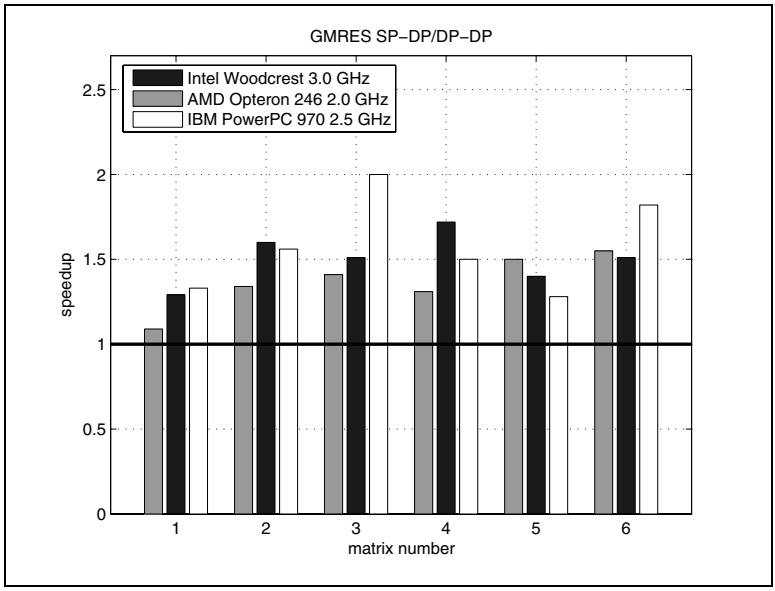


Figure 12. Mixed precision, iterative refinement with FGMRES-GMRES$_{SP}$ from Algorithm 5 *vs.* DP FGMRES-GMRES$_{DP}$.

Figure 13 shows the performance ratio of the mixed precision, inner-outer FGMRES-GMRES$_{SP}$ *vs.* double precision GMRES. This is an experiment that shows that inner-outer type iterative methods may be very competitive compared to their original counter-

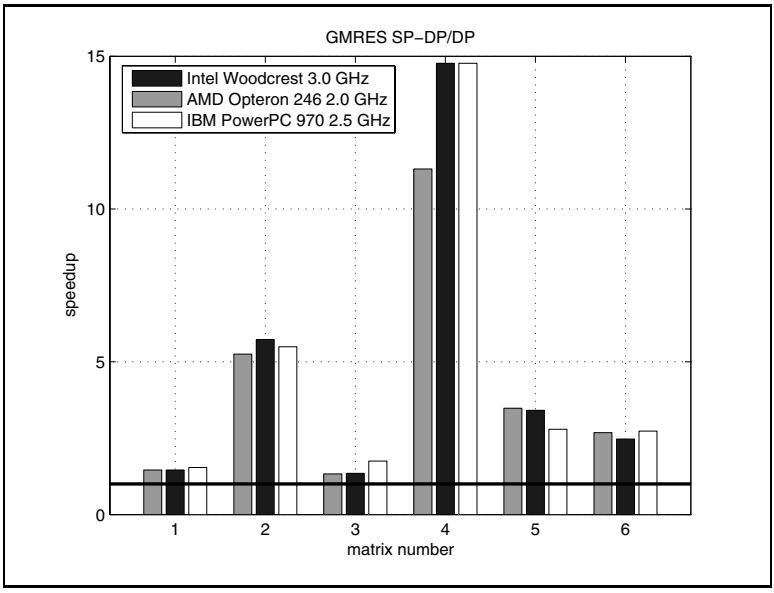


Figure 13. Mixed precision, iterative refinement with FGMRES-GMRES$_{SP}$ from Algorithm 5 *vs* full double precision FGMRES-GMRES$_{DP}$.

parts. For example, we observe a speedup for matrix #4 of up to 15 which is mostly due to an improved convergence of the inner-outer GMRES *vs.* GMRES (e.g., about 9.86 of the 15-fold speedup for matrix # 4 on the IBM PowerPC architecture is due to improved convergence). The portion of the 15-fold speedup that is due exclusively to single *vs.* double precision arithmetic can be seen in Figure 12 (about 1.5 for the IBM PowerPC).

4. Conclusions

The algorithms presented focus solely on two precisions: single and double. We see them however in a broader context of higher and lower precision where, for example, a GPU performs computationally intensive operations in its native 16-bit arithmetic, and consequently the solution is refined using 128-bit arithmetic emulated in software (if necessary). As mentioned before, the limiting factor is conditioning of the system matrix. In fact, an estimate (up to the order of magnitude) of the condition number (often available from previous runs or the physical problem properties) may become an input parameter to an adaptive algorithm that attempts to utilize the fastest hardware available, if its limited precision can guarantee convergence. Also, the methods for sparse eigenvalue problems that result in Lanczos and Arnoldi algorithms are amenable to our techniques, and we would like to study their theoretical and practical challenges.

It should be noted that this process can be applied whenever a Newton or "Newton-like" method is used. That is, whenever we are computing a correction to the solution as in

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

or

$$x_{i+1} - x_i = -\frac{f(x_i)}{f'(x_i)}$$

this approach can be used. We see solving optimization problems as a natural fit.

References

[1] G. W. Stewart. *Introduction to Matrix Computations*. Academic Press, 1973.

[2] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[3] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, 1963.

[4] C. B. Moler. Iterative refinement in floating point. *J. ACM*, 14(2):316–321, 1967.

[5] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 1996.

[6] J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, and J. J. Dongarraa. Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 2006.

[7] J. Kurzak and J. J. Dongarra. Implementation of mixed precision in solving systems of linear equations on the CELL processor. *Concurrency Computat. Pract. Exper.* to appear.

[8] J. Kurzak and J. J. Dongarra. Mixed precision dense linear system solver based on cholesky factorization for the CELL processor. *Concurrency Computat. Pract. Exper.* in preparation.

[9] Iain S. Duff and John K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. 9(3):302–325, September 1983.

[10] Cleve Ashcraft, R. Grimes, J. Lewis, Barry W. Peyton, and Horst Simon. Progress in sparse matrix methods in large sparse linear systems on vector supercomputers. *Intern. J. of Supercomputer Applications*, 1:10–30, 1987.

[11] Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Piotr Luszczek, and Stanmire Tomov. Computations to enhance the performance while achieving the 64-bit accuracy. Technical Report UT-CS-06-584, University of Tennessee Knoxville, November 2006. LAPACK Working Note 180.

[12] Patrick R. Amestoy, Iain S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Eng.*, 184:501–520, 2000.

[13] Patrick R. Amestoy, Iain S. Duff, J.-Y. L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. 23:15–41, 2001.

[14] Patrick R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Comput.*, 32:136–156, 2006.

[15] Timothy A. Davis Iain S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. 18:140–158, 1997.

[16] Timothy A. Davis. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. 25:1–19, 1999.

[17] Timothy A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. 30:196–199, 2004.

[18] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, Jack Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Philadalphia: Society for Industrial and Applied Mathematics., 1994. Also available as postscript file at http://www.netlib.org/templates/Templates.html.

[19] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[20] Kathryn Turner and Homer F. Walker. Efficient high accuracy solutions with gmres(m). *SIAM J. Sci. Stat. Comput.*, 13(3):815–825, 1992.

[21] Gene H. Golub and Qiang Ye. Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM Journal on Scientific Computing*, 21(4):1305–1320, 2000.

[22] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. Technical Report 91-279, Department of Computer Science and Egineering, University of Minnesota, Minneapolis, Minnesota, 1991.

[23] Valeria Simoncini and Daniel B. Szyld. Flexible inner-outer Krylov subspace methods. *SIAM J. Numer. Anal.*, 40(6):2219–2239, 2002.

[24] O. Axelsson and P. S. Vassilevski. A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning. *SIAM J. Matrix Anal. Appl.*, 12(4):625–644, 1991.

[25] Y. Notay. Flexible conjugate gradients. *SIAM Journal on Scientic Computing*, 22:1444–1460, 2000.

[26] C. Vuik. New insights in gmres-like methods with variable preconditioners. *J. Comput. Appl. Math.*, 61(2):189–204, 1995.

[27] J. van den Eshof, G. L. G. Sleijpen, and M .B. van Gijzen. Relaxation strategies for nested Krylov methods. Technical Report TR/PA/03/27, CERFACS, Toulouse, France, 2003.

[28] H. A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, 1994.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

A Model for the Design and Programming of Multi-cores

Chris JESSHOPE
Informatics Institute, University of Amsterdam

Abstract. This paper describes a machine/programming model for the era of multi-core chips. It is derived from the sequential model but replaces sequential composition with concurrent composition at all levels in the program except at the level where the compiler is able to make deterministic decisions on scheduling instructions. These residual sequences of instructions are called microthreads and they are small code fragments that have blocking semantics. Dependencies that would normally be captured by sequential programming are captured in this model using dataflow synchronisation on variables in the contexts of these microthreads. The resulting model provides a foundation for significant advances in computer architecture as well as operating systems and compiler development. The paper takes a high-level perspective on the field of asynchronous distributed systems and comes to the conclusion that dynamic and concurrent models are the only viable solution but that these should not necessarily be visible to the users of the system.

Keywords. Programming models, Concurrency, Multi-cores

Introduction

This paper presents the results of more than ten years of research that started in the mid 1990s and whose goal has been to produce an architectural model that can provide solutions for effectively programming distributed multiprocessor systems. Initially, the goal was to support a programming model based on the data-parallel abstraction [1,2]. Today, the goal posts have moved and our aim is to support the programming of distributed multiprocessor systems using sequential languages, while also supporting other forms of concurrent software engineering, such as data-parallel, streaming and threaded, etc. The architectural landscape has also changed and a whole new generation of engineers and scientists are tackling the challenges of a new era of multi-core chips. The fact is however, that the problems remain unchanged, at least in principal issues, with only a minor shift in a few of the underlying parameters.

The work described here offers systematic solutions that can be applied across design spaces represented by large parametric ranges. This makes it applicable both to multi-cores as well as to so-called Grids. Indeed, in reflection on the period described above, one could argue that it is the Grid bandwagon, an utterly pragmatic diversion of resources that has been responsible for the lack of funding in systematic research in the general area of concurrent computing.

Systematic solutions, that span the entire range parameters from multi-cores to distributed systems, are rooted in the design of processor architectures that are both efficient and can tolerate a large latency in responding to external events, such as

accessing distributed memory or in the communication between distributed processes or threads. To achieve this requires the asynchronous scheduling of some unit of concurrency so that the processor is able to dynamically schedule work in order to hide latency. This use of concurrency to hide latency is called *parallel slackness* [3] and the concept predates this reference by probably another decade.

The first direction our work has focused on was therefore in reducing latency in communication [4], as there is always a cost in supporting parallel slackness. To increase the amount of latency that can be tolerated means providing additional synchronising memory for scheduling the units of concurrency. The more latency to be tolerated the more synchronising memory is required for a given granularity. Synchronising memory in its most general form detects an enabling event (e.g. arrival of remote data) and schedules the unit of concurrency dependent on it.

The second research direction was in being able to manage the finest level of granularity in terms of the frequency and size of the unit of concurrency that was scheduled, as this also has an impact on efficiency and can be best explained by analogy. Were scheduling to use regular units of concurrency with statically known parameters, we could use the analogy of building a wall. Whatever the size of brick, our wall would always contain the same amount of matter as we can arrange regular units without any gaps (this analogy ignores the mortar). However, scheduling is dynamic and the units of concurrency are irregular. A better analogy therefore, is a bucket of sand vs. a bucket of stones. Because they are irregular, the larger the unit, the less matter can we pack into a bucket.

This led us to look at instruction-level scheduling of small units of concurrency and resulted in a proposal for a dynamically scheduled RISC processor [5]. This model of scheduling microthreads in a DRISC processor has been developed over the intervening years and the recent resurgence of interest in architecture has accelerated these results following the rather sudden and recent realisation that superscalar architectures must give way to multi-cores. The result for us is a machine/programming model that gives all of the advantages of our current sequential one, yet can be used to program multi-cores through to the end of silicon scaling with similar properties in terms of binary-code compatibility.

1. Influences on this work

There are a number of significant results that have influenced this work. By far the earliest of these was the pioneering work by Burton Smith. A number of processors he designed have successfully managed units of concurrency at the instruction-level. These included the Delencor HEP [6] in the early 1980s, the Horizon, and culminated in the Tera architecture [7].

The HEP supported multiple blocking processes interleaved on a cycle-by-cycle basis in an eight-stage pipeline. This required at least eight active processes to keep the pipeline full. The HEP used parallel slackness to tolerate latency in accessing its distributed memory, by using a queue of suspended processes that were rescheduled by the arrival of data or a write acknowledgment from memory. The concurrency required in a single processor for accessing memory was estimated to be about 20 processes per processor for a 4-processor system [6]. The HEP supported synchronisation between processes by adding full-empty bits to data in memory, so that a read to an empty location or a write to a full location would suspend the process until the operation

completed. By providing both normal and blocking operations to memory, various programming models could be supported. Delencor exposed this model in HEP-FORTRAN, which added asynchronous variables, which defined synchronisation between processes, and a CREATE construct that was used for creating processes and which mapped onto the machine instruction that implemented it.

The Tera, which later became the basis for Cray's MTA product (The Tera Computer Company bought the remains of the Cray Research division of Silicon Graphics in 2000 and renamed itself Cray Inc.) incrementally improved on the earlier architectures. It could support up to 512 processors and a similar number of I/O and cache processors connected by a sparse 4096-node indirect network implemented as a three-dimensional torroid. Memory was randomised and with the larger number of processors, the latency of memory access in Tera increased to around 70 processor cycles on average [7]. Tera also supported the issue of multiple instructions from a single instruction stream reducing the number of threads required to tolerate the latency by allowing the compiler to statically schedule instructions with known delays. Tera processors still supported 128 streams each with 32 GP registers and 8 branch target registers per stream, giving 4096 GP registers and 1024 target registers per processor. Given that Tera had a mildly wide instruction issue (3 operations per cycle), the 4096 by 9-port GP register file would have been a major implementation issue.

Both HEP and Tera were significant architectural achievements but neither was a commercial success. The problem in both cases was twofold. Firstly Delencore forced a new programming model on the user, when at the same time, vector supercomputers could be programmed using a sequential model, even though code had to be tweaked in order to obtain optimal performance. The advantage of the vector platform was that reasonable performance could be obtained immediately and then by incrementally transforming the code, the vectorising compiler could begin to approach the peak performance. With HEP it was parallel or nothing, as sequential code ran at $1/8^{th}$ of a single processor's peak performance. This fact was exacerbated by the technology used, for example, GaAs logic was used in implementing the HEP, which although fast, was only implemented in small scale integration leading to cost and reliability problems. Only in 1999 did Tera switch to CMOS technology. Indeed, technology seems to have a history of killing off innovative computer solutions, as it was also partly responsible for the demise of the next milestone in this area, the transputer.

The transputer [8] was developed by INMOS and, like the HEP and Tera, supported multiple processes explicitly at the ISA level. In the transputer, concurrency was supported by two instructions in its ISA to start and end a process, as well as the implementation of a read and write instruction on a channel, which could be virtualised. Whereas the HEP supported parallel slackness and scheduling through instructions reading and writing memory, the transputer supported it by reading and writing a channel. Virtualisation was introduced using a static, link-time mapping of processes to processors. If communicating processes are mapped to the same processor, the communication channel is implemented by a memory word rather than a physical link (each transputer had only four physical links). Communication over channels was implemented by two further instructions that implicitly managed scheduling and process suspension was achieved by holding the suspended process identifier in the channel word. Channels therefore provided synchronising communication between processes implemented in a CSP style of programming defined by a language called occam [9].

Similar work was also undertaken in the dataflow community, especially work by Papadopoulos and Culler on the Monsoon architecture [14], which had many similarities with the work by Burton Smith. The difference between their work was that in Monsoon, programs were implemented in the context of dataflow graphs rather than the explicit management of threads of conventional instructions (i.e. instructions target other instructions rather than named locations in the register file). The authors later realised that evaluating simple arithmetic expressions suffered in this implicit model [15]. The work reported here also commenced with a study of dataflow techniques and we too were convinced that exposing concurrency through concurrency management in a conventional RISC processor was the most fruitful direction for efficiency in implementation.

2. Machine/programming models

If we consider machine/programming models generally, we can identify three kinds of model. The first is the *sequential model*, which has been around for some 50 or so years and which has changed very little in that time; the second might be called *ad-hoc parallel*, where a range of limited concurrency primitives are grafted onto the sequential model and finally we have *fully parallel models*, where concurrency is treated as a first-class concept. Perhaps the best way to differentiate between the latter two is to ask whether the model is used to expose the concurrency of the machine or used to capture the concurrency of the problem being solved?

The MTA and transputer provide good examples of these different views. From the Cray web site we can read: "Each MTA processor has **up to** 128 RISC-like hardware threads" (http://www.cray.com/products/programs/mta_2/). Thus, even though virtualisation of the processor supported parallel slackness, the concurrency described was the hardware, a bounded concurrency of 128 threads/processor. The transputer on the other hand described the concurrency of the application, as occam allows the user to create an arbitrary number of processes only weakly bounded, through limitations on memory size.

2.1. Sequential model

Before we discard the sequential model completely, there is much we can learn from it and its many advantages. Perhaps the first is that programmers find it easy to understand the model and to write correct programs. The main issue here is one of determinism. Sequential programs are deterministic and give the same results however long it takes to execute its component parts. Concurrent programs on the other hand must introduce synchronisation. Too much synchronisation induces deadlock and too little introduces non-determinism in the results. The issue of non-determinism in programming models is explored more deeply in a recent paper by Lee [10].

Sequential programs are also safely composable under certain reasonable constraints (e.g. encapsulation of data, pure functions etc.) through the well-understood concept of hierarchy. This is not the case for unconstrained parallel models and it is one of the biggest pitfalls to CSP-based languages, such as occam, where deadlock could be easily induced through composition. It is possible to construct programs from networks of processes that are deadlock free by design but these must conform to certain restricted patterns [11]. When patterns of networks are provably equivalent to

such design patterns it is also possible to design programs in a hierarchical fashion [12]. Ideally however, the constraints should be imposed on the machine or at least the programming model in order to preclude deadlock entirely.

There are a number of other practical advantages of the sequential model. Source code is universally compatible and can be compiled to any target without modification; one cannot say the same for many ad-hoc concurrent models. Moreover, over the last decade, binary-code compatibility has been one of the biggest marketing forces in the computer industry. Enormous resources have been directed at keeping binary code compatible in a sequential ISA, over a range of machine generations, each of which increased the concurrency of execution marginally. There was however, an enormous cost in hardware to support this.

The goal of the work described here is to obtain the same benefits of the sequential model, as described above, but from the perspective of a concurrent programming model. That is a deterministic model that is free from deadlock under concurrent composition and one that is binary compatible across a range of implementations from a single processor to the highest level of concurrency a particular application can support. Finally, the Holy Grail is that such binary code should be relatively easy to derive from existing sequential programs, which is not such a mystery. Prior work on parallelising compilers has had the dual problem of exposing concurrency and then scheduling it. Removing the requirement for the latter by providing a true concurrent model that abstracts scheduling removes the most difficult task from that goal.

2.2. The SVP model

The *SVP* model described here was developed in the AETHER project to define a SANE virtual processor (http://www.aether-ist.org/), where SANE refers to a *self-adaptive network entity*. The goal of the project is to explore self-adaptivity in complex computing systems, which in turn requires dynamic concurrency. The model was based on a substantial amount of prior research that started with DRISC [5] and developed through various ad-hoc concurrency models based on microthreading [13]. SVP however, puts the concurrency at its core and provides a full range of concurrency control mechanisms. In doing so, it subsumes various facets of both programming model and operating system functionality.

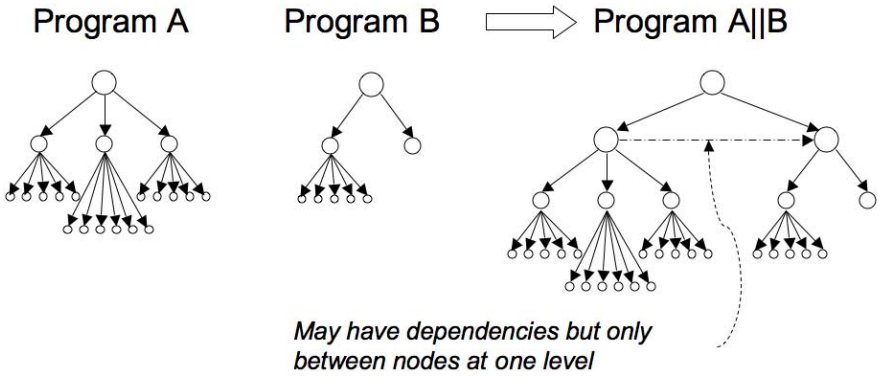


Figure 1. Concurrent composition in SVP

2.3. Composition in the model

In SVP, composition is concurrent and also dynamic. The result of this is that a program in the model can be represented by a dynamically evolving concurrency tree, a snapshot of which is shown in Figure 1. Concurrent composition replaces sequential composition, which is the key to flexibility in terms of implementation granularity. The concurrent composition is based on the sequential model and maps to the constructs found there. Serialisation is therefore trivial and is the mechanism for controlling the granularity of implementation. It is a tacit assumption in what follows, that the leaf nodes are microthreads, i.e. just a few machine instructions sharing a small context of registers. In Figure 1 then, the nodes represent *microthreads* and branching at a node represents concurrent composition. Sequence is introduced into the model in one of two distinct and separate ways:

1. the microthreads comprise sequences of one or more machine instructions. In an implementation, complexity of the nodes at the lowest levels depends on the parameters of the target architecture. In a microthreaded microprocessor, this may only be a couple of instructions, due to its efficient management of concurrency. It should be noted however, that a leaf node could also be a complete binary program that has no understanding of the model at all.
2. explicit dependencies are allowed between microthreads but in a very restricted manner. The dependencies allowed are between the creating thread and its subordinate threads and between all subordinate threads at one level. These are defined as an acyclic dependency chain from the creating thread through each thread created in some defined order.

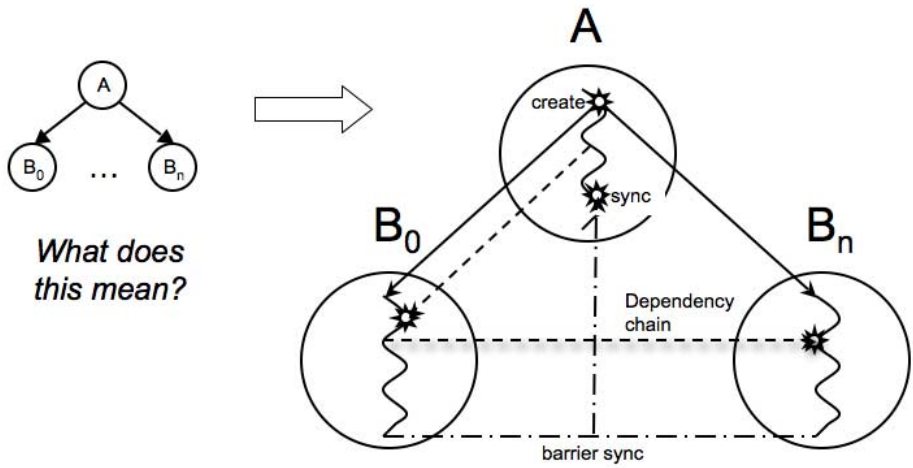


Figure 2. The events in creating a family of threads

Concurrency is introduced in the model through the dynamic creation of a parameterised *family* of microthreads; defined on a single thread definition, see Figure 2. The creating thread A executes a *create* action, which causes an ordered set of threads to be created subject to resource constraints, i.e. not all of the threads need be created simultaneously although they may be. The set may also be semi-infinite, so that an unbounded number of threads may be defined, executed on finite set of resources

and terminated dynamically. The second event in the creating thread A is when all threads in the family have terminated and is identified by a *sync* action. A proceeds asynchronously with B_i but must wait on the *sync* action associated with any family of threads that it creates. This is not optional in most circumstances, as the family Bi cannot reliably communicate any information back to the creating thread without this action. The exception is where the created family is a continuation of the current thread and in this case it must be a family comprising a single thread. Synchronisation is described in more detail in Section 4.

Create is a concurrent analogue to both function invocation and loop structure in the sequential model. The parameterisation of the family captures both static and dynamic loop bounds. As described above, a family is defined as an ordered set and this set is defined over a sequence of index values captured by three parameters: {*start*, *limit*, *step*}. Each thread in the set has available to it, through the implementation of *create*, a unique value from this implied sequence. Dynamic loop bounds are obtained by setting limit to infinity (or as close to this as is possible in a finite machine) and terminating the family from any of the threads created using a *break* action. Note that if SVP is serialised by executing each thread to completion in index sequence, the analogue above is exact.

The use of the family in creating concurrency is in order to amortise the implementation overhead. As already indicated, at the lowest level in the concurrency tree, threads will comprise just a few basic machine instructions and creating them one at a time would cause a significant overhead.

2.4. Communication in the model

Threads in SVP are blocking and can participate in synchronising communication. This communication is deliberately restricted and is implemented as a blocking read or dataflow-like synchronisation. In future silicon systems, communication and synchronisation should be localised and must be exposed to the compiler in some way, as communication costs across chip will be expensive. For this reason we assume that synchronising communication within a family of threads is implemented locally in the machine's registers (or some form of synchronising memory close to the processor).

SVP communication is fine grain (i.e. on scalar values) and where required, defines a set of events that cascade through the set of threads in a created family. The creating thread, A in Figure 2, may write data only to the first thread created in the family, B_0, which will block on the first action requiring that data. Thread B_0 can in turn write to the next thread in index sequence and so on, so that each thread is able to write to the next in index sequence until we get to B_n. To provide closure to this sequence of communications, corresponding data written by B_n is available to the creating thread, A, but this is not a synchronising communication. To maintain symmetry and to enable concurrency controls to be implemented efficiently, this data overwrites the same variable that initiated the dependency chain and is only defined on the *sync* for the family.

It has been suggested that this restricted communication, equivalent to a loop with a loop-carried dependency from one iteration to the next, is too restricted. To put this criticism in perspective however, any regular dependency over an iteration space with a skip distance of greater than one can be transformed by nesting families, so that an independent family of extent equal to the skip distance is based on a thread that creates a subordinate family containing a local dependency. This is illustrated in Figure 3 in

terms of adjacency; it is a transformation on the index space from one to two dimensions.

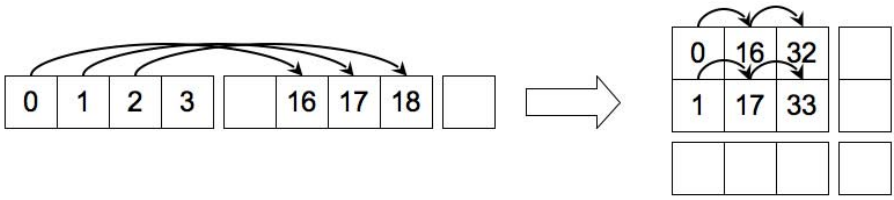


Figure 3. Transforming an index space to achieve local communication of dependencies.

Another technique that can be used to transform non-local dependencies to local ones is static routing between threads in a family. This is achieved by defining scalar buffers in each thread and copying values from one buffer to the next so that each buffer represents a skip distance from one up to the number of buffers used. In a fine grain model, this compiles to a register-to-register copy and is very efficient. The technique is illustrated in Figure 4. Finally, dynamic, index-set transformations can be achieved by data copying in memory, exactly as would be achieved in a vector machine, i.e. gather and scatter operations. However this could be inefficient in some distributed implementations.

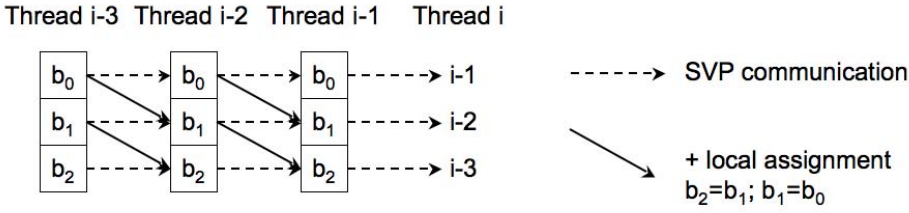


Figure 4. Static routing between buffers in a thread to achieve non-local dependencies

There are very good reasons for this restriction on communication in SVP. It guarantees freedom from communication deadlock within a family. It exposes local communication to the compiler and finally, it allows the compiler to analyse resource deadlock in the case of bounded recursion of families of threads.

There are two other forms of communication in the SVP model. These occur at a higher level of abstraction (and of granularity) and are implicit rather than explicit. The first is that the model assumes a shared memory abstraction and while this may or may not be a basis for implementation, there will certainly be communication involved in any implementation. For example, coherence protocols in a distributed-shared memory or direct communication between distributed memories. The final form of communication is in the implementation of binding a unit of work to a resource, which is described in more detail in section 6.

2.5. Concurrency control

To provide closure on concurrent composition within the model, some form of concurrency management is required. This has far reaching implication in an implementation but is a necessity in the context of implementing self-adaptation in the

AETHER project. In a retrospective on the Monsoon architecture, Papadopoulos and Culler first reiterated the need for concurrency at the machine level: *"Threads are a key agent within all modern machines, and yet there are no operations defined on them at the machine level. Thread operations, such as create and terminate are implemented in software by the operating system..."* and also explored this further: *"Imagine in a conventional instruction set architecture that the register reservation bits are exposed in the instruction set, so it would be possible to branch on the result of a load being 'not yet present'."* Although I do not believe the solution they propose is at an appropriate level of abstraction for the explicit management of concurrency, the notion of reflection on the state of a concurrent computation must be introduced. In SVP, both creation and reflection are supported as explicit actions in the model, which in turn are implemented in the ISA of a microprocessor.

We have already encountered the *create* action and normal termination identified collectively by the *sync* action, possibly initiated by a *break* action in one of the threads. To this we add two further reflective actions on families of threads, namely *kill* and *squeeze*. *Kill* terminates a complete family of threads losing all the state of that computation, whereas *squeeze* terminates a family by identifying and capturing an intermediate state, so that the family can be recreated and completed, probably on different resources. These two actions are executed, not in the creating thread or any of the threads it creates, but in an independent and asynchronous control thread that is monitoring some aspect of the computation or its environment (see Figure 5). These actions require that families must be identified within the machine. It may seem that this is also required for the *sync* action but this action is internal to the creating thread and can be implemented by allocating a local register to receive a return code from the *create* action, which is blocking.

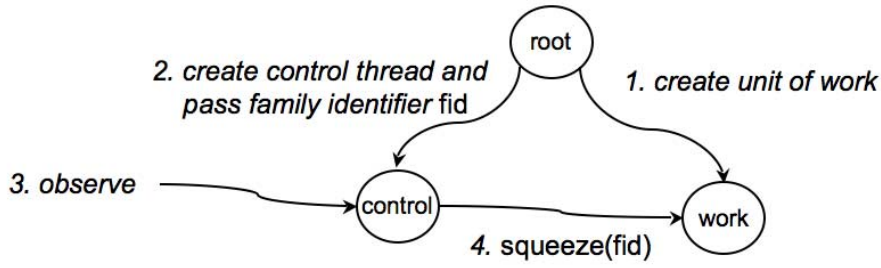


Figure 5. Reflection in the model by named families of threads and concurrency control

Managing concurrency asynchronously requires that any reflective action be observed. Typically this would be in the creating thread, for example root in Figure 5. In SVP this is achieved by providing a return code and possibly a return value to the creating thread via the *sync* action. The return code identifies how a family was terminated, it may be one of the following codes: {*normal*, *break*, *kill squeeze*}, where *normal* is the result of the execution to completion of all threads specified in the *create*, and *break, squeeze* and *kill* indicate that the family was terminated by the action with the same name. A return value is set only by *break* and *squeeze*. In the former it is set by the *break* action in the thread that succeeds in executing its *break* and in the latter it captures a unique squeeze point in the index sequence and is described in more detail below.

One of the far-reaching implications of reflection is the fact that such actions are extremely powerful, especially when implemented at the level of instructions in an ISA. Typically these would be controlled by some user identity in an operating system. I.e. only the user of root is allowed to terminate a job in typical operating systems. Here we have a self-similar model and the capability to implement these actions must be granted to a thread. Only in this way can they be implemented in a manner that can be guaranteed to be secure. In SVP, the family identifier contains a capability, for example a random token generated on creation, which is passed securely to a controlling thread and which must be matched before *kill* and *squeeze* actions are executed. In SVP, a secure channel may be implemented using memory and appropriate memory protection domains or using a synchronised communication, which passes a value by a shared register accessible only to the creating and control threads.

Another issue that must be considered in implementing these reflective actions is whether to implement them as deep or shallow actions, i.e. whether they are propagated down the concurrency tree illustrated in Figure 1 or not. For *kill* it does not make sense to implement the action shallowly, so a *kill* should be propagated from one family to its subordinate families. For *squeeze* the appropriate implementation is not clear. For threads that perform computation it may not be efficient to preempt them, as this may cause as much work in preserving the partial state as in completing the computation itself. However, for control threads that *create* a significant amount of work in their sub trees, some depth of action may be required in order minimise the latency of the action.

The solution adopted in SVP is to allow for a program-controlled depth of propagation of the *squeeze* action. Threads may take on the attribute *squeezable*, which implies that a family defined on this thread is able to propagate any *squeeze* action it receives to its subordinate families. It is the responsibility of the thread to capture any state required to re-create the subordinate thread when re-executed itself. Because of the restriction on communication, the state is limited to one or more scalar values communicated between threads so long as the squeeze identifies a partition on the family, the *squeeze point,* which must also be captured as an index value.

Squeeze returns this index value via the *sync* action in the creating thread. The value is defined as the first thread in index sequence not yet executed to completion. This is the first thread not to have been allocated resources if the thread is not *squeezable.* Alternatively*,* if it is *squeezable,* it will have created subordinate threads, which were also squeezed. Thus any thread that terminates with a *squeeze* return code will have to be re-executed. The result on a family receiving the *squeeze* action is to cease the creation of new threads and allow all currently active threads to complete. In addition, the family will send a *squeeze* action to any subordinate families created by its threads but only if the thread is labeled as being *squeezable*.

The algorithm to determine the index value therefore depends on whether the thread is *squeezable* or not. If the thread is *squeezable*, the index value captured by *sync* is defined as the index of the first thread in index sequence to have been allocated a context of synchronsing memory but which has not yet terminated normally. This index partitions the family into those threads that have completed and those threads must be re-executed because they created a subordinate family and that family was squeezed. A *squeezable* thread must update the start index of the *create* parameters of its subordinate family with the return value at *sync* so that on re-execution it will *create* only those subordinate threads that had not completed.

Figure 6 illustrates three partitions over the ordered set of threads in a family on receiving a *squeeze* action. The first, identifies those threads that have all completed, these are not re-executed. The second, those threads that have been allocated contexts and which may already have completed, these are allowed to complete, if necessary by squeezing their subordinate families, but they will all be re-executed when the family is re-created; this partition is identified by at least one thread (the first) that has not yet completed. Finally, there are those threads in the family not yet allocated a context of synchronising memory, which will of course need re-execution when the family is re-created.

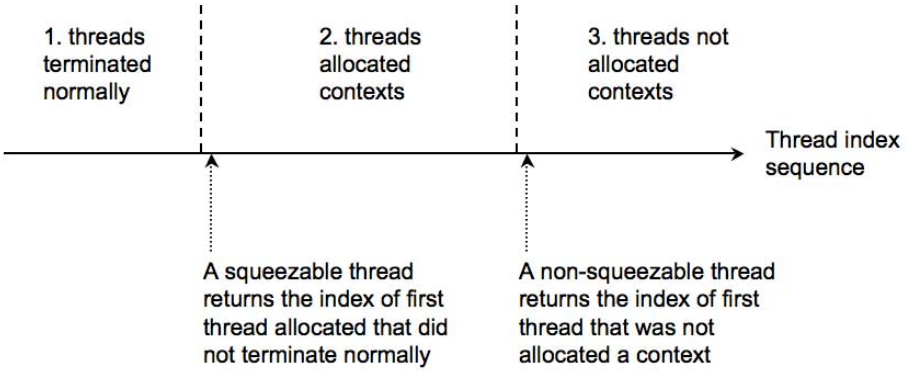


Figure 6. Determining the squeeze point in an executing family of threads

If the family is not squeezable it will not pass the *squeeze* action onto any subordinate family and all threads in partition 2 in Figure 5 will eventually terminate normally and the return value at *sync* is set to the index of the first thread not to have been allocated a context of synchronising memory.

On synchronisation, the value of any dependency at the *squeeze* return index is automatically saved to the variable that initialised the dependency chain in the creating thread, again allowing the family to be re-created from the *squeeze* index position to complete it.

Squeezing a family of threads allows the task defined by that family to be preempted and restarted at a different *place* (set of processors). By using the concept of a squeezable thread a disciplined approach to the rapid preemption of a concurrently executing program can be provided in the model. This concept is at the heart of the model and provides one of the most important features for implementing self-adaptivity.

2.6. Summary of the model

The SVP model provides a number of actions that initiate and control concurrency. These actions are summarised below and captured schematically in Figure 7.

- *create* - defines a parameterised family of microthreads, which are allocated contexts for execution as resources become available. Microthreads may themselves *create* subordinate families of microthreads;
- *sync* - identifies the termination of all threads in a particular family. Using a *create*/*sync* pair allows the creating thread to continue asynchronously with

the threads it creates. *Create* returns a code via its *sync,* which defines how the family terminated. If termination was by a *break* or *squeeze* action the *sync* also provides a return value;

- *break* - terminates a family of threads on a condition being met within one of its threads, the thread successfully executing a *break* will set the return value;
- *kill* - terminates an identified family of threads so that no state set by that family can be guaranteed. The *kill* action is propagated to all subordinate families;
- *squeeze* - terminates a family of threads so that its state can be captured and the family can be re-executed. *Squeeze* is preemption over a concurrency tree and returns an index value of the first thread to be executed when the family is recreated. A *squeeze* action is only propagated to a subordinate family of threads if those threads are defined as being *squeezable*.

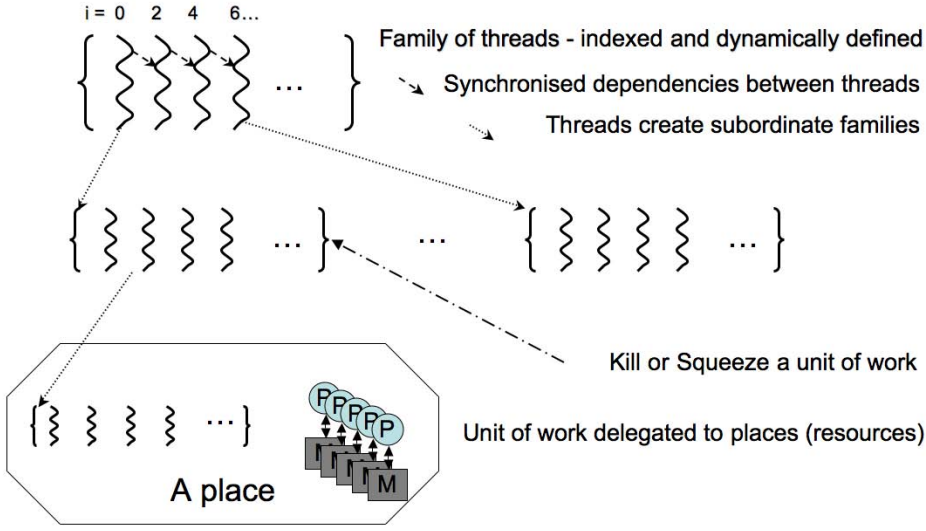


Figure 7. A schematic summary of the SVP model

It should be emphasised that SVP is a dynamic model that creates threads dynamically on resource availability. Both the extent of the family and the resources that the family are executed on are defined when the *create* action is executed. The *create* action captures all parameterisation within the model; it defines:

- the threads' code - possibly in many different forms (e.g. specialised implementations on reconfigurable logic);
- the family's parameters, i.e. extent and index range;
- whether the family has dependencies between threads;
- the extent of concurrency of a unit of work - can be inferred from the above two facets applied recursively;
- where the unit of work is to execute - i.e. the delegation of a family to some place or resource set;
- whether the *create* requires mutual exclusion or not where it is executed;
- the notion of a timer, if required for partial failure;
- any other meta-data required, e.g. real-time constraints, power constraints etc.

3. SVP memory model

The state of the SVP model at any given time is defined by two abstractions, the first and most persistent is an asynchronous shared memory, which comprises a number of addressed locations, each of which may be read and written to by a thread but subject to certain constraints. The constraints are due to the asynchronous and concurrent update of the memory by multiple threads. No guarantees can be given about the access time to this memory. The second abstraction is a context of syncronising memory associated with each thread, which provides the operands to all arithmetic or logical operations in the thread. It provides synchronisation on scalar values produced by other threads in the same family or from values input from shared memory.

3.1. Shared memory

SVP is based on units of work, which comprise families of threads (and their subordinate threads). A family has inputs and outputs, which are defined by the thread used to *create* it. These inputs and outputs are the locations in shared memory that the threads read and write. The output can only be defined on the *sync* for a family, i.e. the memory is bulk synchronous. Prior to this event, because we are dealing with a potentially asynchronous concurrent system, there will always be a subset of locations whose state cannot be determined. This means that in order to give deterministic programs, no two concurrent threads may read after write to the same location of memory and no two concurrent threads may write to the same location.

Note that this abstraction of shared memory may be implemented in any manner including using distributed memories and message passing between them. In the latter case, before a family of threads is created at a place, a synchronising communication must provide the place with the data that the family will read as input and before the family can be deemed to have synchronised, all of its output must be returned. Figures 8 and 9 show the shared and distributed view of family creation.

3.2. Synchronising memory

Synchronising memory provides the mechanism by which threads within a family can synchronise with each other and their creating thread; it also provides the mechanism for suspending and rescheduling threads. All operands must be loaded into this memory before the processor can perform the corresponding operations. It therefore provides the mechanism by which the processors can synchronise with asynchronous-shared memory.

Synchronising memory is dynamic and transient. An implementation dynamically provides a context of scalar variables for a thread, which are allocated empty, written once identifying a synchronising event and which are discarded when the thread completes. Each location provides a blocking read or dataflow-like synchronisation and dependencies between operations in different threads are enforced by this synchronising memory. It is assumed that this memory is fast, distributed and “close” to the processor or logic cells implementing the thread.

Note that threads may be sequentialised on their index order if synchronising memory is limited. So, during the execution of a family, there may only be a subset of the family’s threads active and contributing to this synchronising state. This constrains thread creation to be performed in index order, at least for dependent families of

threads. The amount of synchronising memory allocated will limit the parallel slackness and hence extent of latency that can be tolerated.

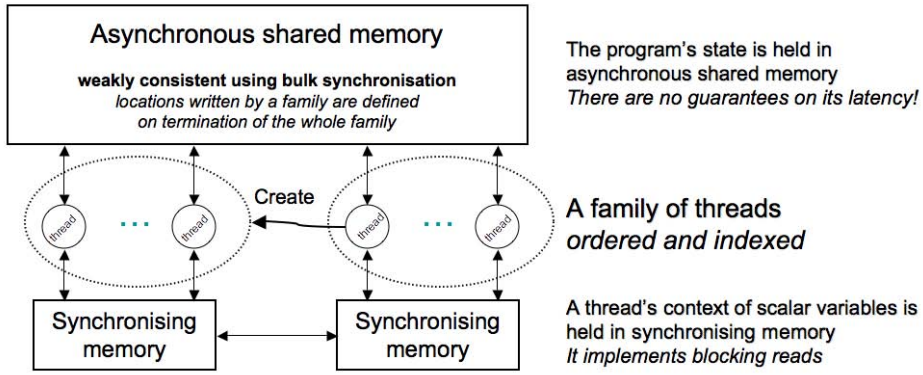


Figure 8. Family creation in a shared-memory environment one thread creates a new family of threads.

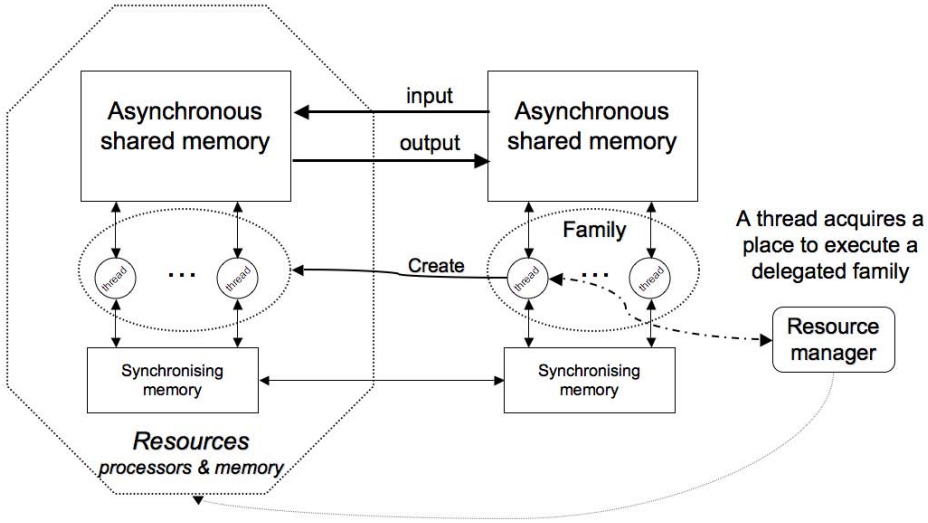


Figure 9. Family creation showing the dynamic resource management required with SVP. Again one thread creates a new family but this time it acquires a new place to execute the family. Here a place is a set of processors and associated distributed memory.

4. Resources in the model

Programming models are abstract but when implemented they have to deal with physical entities, i.e. resources such as processors and memory, access to hardware such as I/O registers and security issues such as access controls. SVP binds resources dynamically by creating a *family* at a *place* where the threads will be executed. Typically a place will be one or more processors, an accelerator, FPGA cells etc. This process of binding a family to a place is called *delegation* and is like a remote-procedure call, where the "procedure" is the *unit of work* defined by the family being

created and any subordinate families that it may *create*. Like the model itself, delegation is hierarchical and a unit of work can delegate one of its subordinate families to yet another place.

To make the model amenable to static analysis, regardless of the fact that the number of processors and extent of the family may be dynamic, the distribution of threads to processors on family creation is deterministic. This will be implementation dependent and will typically be a cyclic or block-cyclic distribution. Thus, if each processor at a place has access to:

- the number of processors in the ordered set of processor allocated for a delegation;
- its own location in the set;
- the parameters defining the set of threads, {*start, limit, step*};

then each processor can independently allocate only those threads that comprise its distribution.

One of the key parameters in family creation is the number of threads created at once on a single processor, called the *block* size. This parameter can be used to manage resources and avoid resource deadlock at a place, at least this is possible with bounded recursion of *creates*.

Note that the concept of place must deal with more than processor resources; it must also deal with security and system services. A place will be identified by some namespace (address) through which the creating thread can communicate parameters to the processors for thread creation. For security, there must be a capability associated with a place, so that an arbitrary thread cannot interfere with any legitimate families executing. Again this can be a random token given to the place and creating thread on resource allocation, which is then matched on delegation.

Creating at the *default* place executes the family on the same resources that are being used to execute the creating family of threads. Another identified place is *local,* which forces a family to execute all of its threads on the same processor as the creating thread. This allows for control of locality. A local family may only create parallel slackness.

More generally, a place is an abstraction, which gives a union of:

- a communication address defining a physical place to which the family's parameters are sent on delegation;
- a capability to execute the family at that place;
- optionally - a service provided at a particular place, which requires exclusion.

4.1. Resource deadlock

Although the model is designed to be free of communication deadlock by design, like all data-driven models, it is still susceptible to resource deadlock. However, resource deadlock is easier to manage than communication deadlock and can be analysed and controlled.

Resource deadlock occurs through the unfettered exposure of concurrency with dependencies on a finite set of resources. As SVP restricts dependencies between threads, i.e. communication between adjacent threads in index (creation) order, then resource usage in a family, i.e. size of the contexts of synchronising memory required to avoid deadlock can be statically analysed; the minimum resource required is the sum of one producer's and one consumer's contexts.

However, in SVP there are also dependencies between a thread and any families of threads that it creates. With finite resources and unbounded recursion of creates resource deadlock cannot be avoided. This is illustrated in Figure 10. When synchronising memory is exhausted, no family can be created and any failed *create* will block the creating thread from completing and so on back up to the root thread.

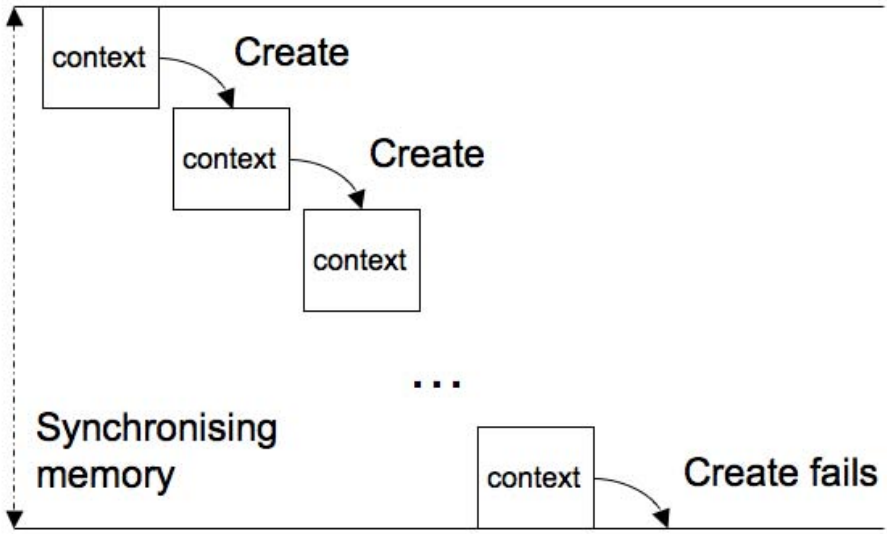


Figure 10. Resource deadlock caused by the limitation of synchronising memory with unbounded recursion of creates. No thread can release its resources until its subordinate families have completed. When synchronising memory runs out no thread can complete.

This situation can always be identified although it does not always indicate deadlock. However, with constraints on the way units of work are allocated, deadlock can be detected, e.g. the example in Figure 10. Even more generally, given unbounded recursion, the weaker signal of possible deadlock can be used to trigger resource acquisition and to resolve the potential problem by delegation. Note that this is conceptually no different to stack overflow in the sequential model. The difference is that here, it is synchronising memory rather than main memory that is the limitation.

5. Implementation of the model

SVP can be implemented at a number of different levels of granularity. The key issue is at what level in the concurrency tree are units of work distributed and synchronisations achieved. For example a coarse-grain implementation could be implemented on distributed systems of conventional processors, provided that any units of work below the distribution level are serialised in order to amortise the communication and synchronisation overheads.

At the University of Amsterdam, we are interested in very fine-grain implementations, where the model is captured in the ISA of a microthreaded processor. We have implemented prior microthreaded processors that supported a single level family via software emulation [16,17], as well publishing estimates of the area required for the support structures required in a silicon implementation [18]. During the last year

we have implemented a further software emulation of the model described here based on the Alpha ISA. The only additional support structures required for the hierarchical model is a table to store family-related information. However, the implementation also requires some re-partitioning of thread-related information, such as queues of active threads, waiting threads etc, as well as requiring family identifiers to be propagated through to the load/store unit in the pipeline for synchronisation purposes. Results on this work will be published soon.

The model has also been captured as an extension to the C language called μTC and we are developing a compiler from this intermediate language to target the emulation of the Alpha-based DRISC microprocessor [20]. Another development we have started is the compilation of C to μTC. Although many parallelising compilers have been attempted, we believe that the lack of success in generating commercial interest in this approach is not the exposure of concurrency in the compiler. Rather it is the requirement to statically schedule the concurrency exposed. With SVP this is no longer a requirement, as the parallelising compiler can simply express all concurrency exposed and leave the scheduling and resource management to run-time systems, in the case of a DRISC implementation, implemented in the instruction set.

6. Conclusions

This paper has summarised the results of a significant research effort carried out over the last ten years. The culmination of this research is a hierarchical model of concurrency that has dynamic data-driven scheduling as well as dynamic management of resources. It can be argued that as systems become more and more complex, i.e. with the introduction of more and more processors servicing a user task, then it is no longer possible to statically analyse and schedule the execution of that task.

The model has all of the attributes of the sequential model. It is deterministic; its is deadlock free by construction and most importantly, when code is compiled to processors that support this model at the instruction level, then that binary code is binary compatible across any number of processors up the limits of concurrency dictated by the application itself. This is a significant property and one that should provide serious consideration of this model.

Finally we believe that it is possible when using this model to be able to compile from sequential code and thus realise the holy grail of being able to take existing sequential code bases and distribute these across distributed systems, be they fine-grain implementations based on DRISC processor or coarse-grain implementations based on existing distributed environments, such as we have in today's globally networked world.

We do not underestimate the difficulties in achieving these goals. The model is disruptive in its efforts to push the management of concurrency into the lowest levels of granularity in processing systems, i.e. at the level of basic machine instructions. These instructions usurp operating system functionality and turn the current perception of operating systems on their head. Currently operating systems apply the principle that processor cycles are expensive and share these cycles between user tasks. Of course concurrency is required on a single processor in the form of parallel slackness but this concurrency should only be used to tolerate latency and should be derived from single user task. It is OK for a processor to be idle, in the world of multi-cores, what is not acceptable is for idle processors to be drawing power. Data-driven rather than

speculative models also provide the necessary mechanisms to recognise this property. For example the DRISC processor measures work not by the number of created processes but by the number of threads in its active queue. The threads will still exist in synchronsing memory but may be inactive waiting for long latency events.

In a modern operating system, on-chip storage and locality of communication are far more important issues and the operating system should be optimising these rather than doling out processor cycles. SVP therefore will have an impact on compilers, operating systems and indeed middle-ware in current distributed systems. These are indeed exciting times to be working in the area of computer systems architecture!

7. Acknowledgments

The work described in this paper is currently being supported by two projects. The first, *Microgrids*, is a national project in the Netherlands funded by NWO and the second, *AETHER*, is a collaborative project funded by the European Community. We gratefully acknowledge the financial support received from both projects, without which the significant progress made over the last year in the development of the SVP model would not have been possible.

References

[1] V B Muchnick and A B Shafarenko (1996) *Data Parallel Computing - the Language Dimension*, ISBN: 1 85032 179 5, Chapman Hall.

[2] C R Jesshope (1992) The f-code abstract machine and its implementation, *Proc COMPEURO 92* (IEEE Press) pp 169-174

[3] G V Wilson (1993) A Glossary of Parallel Computing Terminology, *IEEE Parallel & Distributed Technology*, **1**(1), pp 52-67

[4] C R Jesshope and C Izu (1993) The MP1 network chip and its application to parallel computers, *The Computer Journal,* **36**, pp763-777

[5] A Bolychevsky, C R Jesshope and V B Muchnick, (1996) Dynamic scheduling in RISC architectures, *IEE Trans. E, Computers and Digital Techniques*, **143**, pp 309-317.

[6] J W Moore (1983) The HEP Parallel Processor, *Los Alamos Science*, Fall 1983, pp 72-75. http://library.lanl.gov/cgi-bin/getfile?09-04.pdf

[7] Alverson, R. et al. (1990) The Tera Computer System, Proc. of the 4th International Conference on Supercomputing, Amsterdam, The Netherlands, 11-15 June, pp. 1-6. ACM Press, New York, NY, USA

[8] D May and R Shepherd (1984) The transputer implementation of occam, *Proc. Intl Conf on Fifth-Generation Computer Systems*, Tokyo, pp533-541.

[9] INMOS (1984) occam Programming Manual, Prentice-Hall, ISBN 0-13-629296-8

[10] E lee (2006) The problem with threads, *IEEE Computer*, **36**(5), pp. 33-42

[11] P H Welch, G R R Justo and C J Willcock (1993) Higher-Level Paradigms for Deadlock-Free High-Performance Systems,*Transputer Applications and Systems '93*, Proceedings of the 1993 World Transputer Congress, **2**, pp 981–1004, ISBN 90-5199-140-1.

[12] J.M.R. Martin (1996) *The Design and Construction of Deadlock-Free Concurrent Systems*, PhD Thesis, University of Buckingham, UK, 1996 http://wotug.ukc.ac.uk/parallel/theory/formal/csp/jeremy-martin/

[13] Jesshope C. R. (2006) Microthreading a model for distributed instruction-level concurrency, *Parallel processing Letters,* **16**(2), pp209-228, ISSN: 0129-6264

[14] G M Papadopoulos and D E Culler (1990) Monsoon: An Explicit Token Store Architecture, *Proc. 17th International Symposium on Computer Architecture*, pp82-91

[15] G M Papadopoulos and D E Culler (1998) Retrospective: Monsoon: An Explicit Token Store Architecture, In *25 Years of the International Symposia on Computer Architecture: Selected Papers*, pp. 74--76

[16] Kostas Bousias and Chris Jesshope(2005) The Challenges of Massive On-Chip Concurrency, *Lecture Notes in Computer Science*, **Volume 3740**, pp157 - 170

[17] Bousias, K, Hasasneh N M and Jesshope C R (2006) Instruction-level parallelism through microthreading - a scalable Approach to chip multiprocessors, *BCS Computer Journal*, **49** (2), pp 211-233

[18] Bell, I, Hasaasneh, N and Jesshope C R (2006) Supporting Microthread Scheduling and Synchronisation in CMPs. *Intl. J Parallel Processing*, Jan 2006, pp1-9, DOI

[19] Jesshope, C R (2006) μTC – an intermediate language for programming chip multiprocessors, *Proc. Pacific Computer Systems Architecture Conference 2006 - ACSAC06*, ISBN 3-540-4005, **LNCS 4186**, pp147-160

[20] T. Bernard, K. Bousias, B. de Geus, M. Lankamp, L. Zhang, A. Pimentel, P.M.W. Knijnenburg, and C.R. Jesshope (2006) A Microthreaded Architecture and its Compiler, *Proc. 12th International Workshop on Compilers for Parallel Computers (CPC)*, M. Arenez, R. Doallo, B.B. Fraguela, and J. Tourino (eds.), pp 326-340

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Pegasus and DAGMan From Concept to Execution: Mapping Scientific Workflows onto Today's Cyberinfrastructure

Ewa Deelman[1], Miron Livny[2], Gaurang Mehta[1], Andy Pavlo[2], Gurmeet Singh[1], Mei-Hui Su[1], Karan Vahi[1], R. Kent Wenger[2]
[1]*USC Information Sciences Institute, Marina Del Rey, CA 90292*
[2] *University of Wisconsin Madison, Madison, WI 53706*

Abstract: In this chapter we describe an end-to-end workflow management system that enables scientists to describe their large-scale analysis in abstract terms, then maps and executes the workflows in an efficient and reliable manner on distributed resources. We describe Pegasus and DAGMan and various workflow restructuring and optimizations they perform and demonstrate the scalability and reliability of the approach using applications from astronomy, gravitational-wave physics, and earthquake science.

Keywords: workflow management, workflow optimization, abstract workflow, workflow provenance, cyberinfrastucture

Introduction

Scientific workflows are becoming an enabler of complex scientific analyses. They provide a representation of complex analyses composed of heterogeneous models designed by groups of scientists. At the same time, workflows have also become a useful representation that is used to manage the execution of large-scale computations. This representation not only facilitates overall creation and management of the computation but also builds a foundation upon which results can be validated and shared. Since workflows formally describe the sequence of computational and data management tasks, it is easy to trace back how particular data were derived. Workflows have also become a tool capable of bringing sophisticated analysis to a broad range of users, enhancing scientific collaboration and education. There are many workflow systems currently being developed [7]. In this chapter we concentrate on two complimentary systems Pegasus [28, 31] and DAGMan [22, 34] that together enable efficient and robust execution of large-scale scientific workflows in distributed environments.

In order to facilitate workflow creation, scientists need to be allowed to formulate the workflows in a way that is meaningful to them using high-level abstractions that specify the overall structure of the analysis and the data to be operated on (via a visual or textual interface) in a resource-independent way. This abstract workflow (or workflow instance) is important because it uniquely identifies the analysis to be conducted at the application level without including operational details of the execution

environment. The instance can thus be published along with the results to describe how a particular data product was obtained.

In order to support the abstract workflow specifications which let scientists concentrate on the science rather than on the operational aspects of the cyberinfrastructure (such as the Open Science Grid [2] or the TeraGrid [1]), mapping technologies are needed to automatically interpret and map the user-defined workflows onto the available resources. This is an approach analogous to traditional computer programming methods, where high-level languages are used to describe the computation without needing to specify the use of specific registers or memory locations. In this analogy, the "workflow mapping engine" is a compiler that translates between the high-level specifications and the underlying execution system and optimizing the executables based on the target architecture. The mapping includes finding the appropriate software and computational resources where the execution can take place as well as finding copies of the data indicated in the workflow instance. The mapping process can also involve workflow restructuring geared towards optimizing the overall workflow performance as well as workflow transformation geared towards data management and provenance information generation. The mapping process is usually automated and in our work it is done by Pegasus [28, 31].

The result of the mapping process is an executable workflow, which can be executed by a workflow engine that follows the dependencies defined in the workflow and executes the activities defined in the workflow nodes. DAGMan [35], our workflow engine relies on the resources (compute, storage, and network) defined in the workflow to perform the necessary actions. As part of the execution, data are generated along with their associated metadata and any provenance information that is collected.

The separation of concerns between workflow generation, workflow mapping, and workflow execution allows us to design software in a modular way and to optimize the components based on their functionality. Additionally it allows us to interface to a variety of workflow generation systems.

In this chapter we describe Pegasus and DAGMan and illustrate the workflow optimizations and restructuring techniques and their use in large-scale scientific applications.

1. Pegasus and DAGMan

Currently, Pegasus takes a workflow description in a form of a Directed Acyclic Graph in XML format (DAX). We recognize that not every scientist would be willing to write workflows in XML (or write scripts to generate XML), and that users may want to use simple languages or point and click GUIs. Thus we have integrated Pegasus into a variety of workflow instance generation systems such as VDL (Virtual Data Language) [33], Wings [36], Triana [52], and most recently we have developed a prototype integration with Kepler [44].

Of particular interest is the integration with Wings [37]. Wings uses rich semantic descriptions of components and workflow templates expressed in terms of domain ontologies and constraints. Wings has a workflow template editor to compose components and their data flow. Wings assists the user by enforcing the constraints specified for the workflow components. Wings also helps with data selection to ensure the datasets selected conform to the requirements of the workflow template. With this

information, Wings generates a workflow instance that specifies the computations (but not where they will take place) and the new data products. For all the new data products, it generates metadata attributes by propagating metadata from the input data through the descriptions and constraints specified for each of the components.

In some cases, scientific applications want to provide users with an interface, which is only in the form of a metadata query. For example, in astronomy, users often do not want to know the details of the underlying system, instead they want to retrieve images of an area of the sky of interest to them. In such cases Pegasus is usually integrated into a portal environment where the user is presented with a web form to fill in the desired metadata attributes. Inside the portal, the workflow instance is generated automatically based on the user's input and is given to Pegasus for mapping and then to DAGMan for execution [50]. Examples of this approach can be seen in the Montage project (an astronomy application) [18, 40], the Telescience portal (a neuroscience application)[42], and the Earthworks portal (an earthquake science application) [46]. In all these applications, Pegasus and DAGMan are being used to run the application workflows on national infrastructure such as the TeraGrid.

Pegasus can map workflows onto a variety of target resources such as those managed by PBS [39], LSF [55], Condor [32], and individual machines. Figure 1 shows an overview of the workflow generation and execution process. The executable workflow produced by Pegasus has directives to DAGMan for the execution of the workflow components. These directives include remote job execution, data movement, and data registration. Authentication to remote resources is done via GSI [54]. During the workflow execution, we capture provenance information about the execution tasks. Provenance includes a variety of information such the hosts where the tasks have executed, the runtime, environment variables, etc.

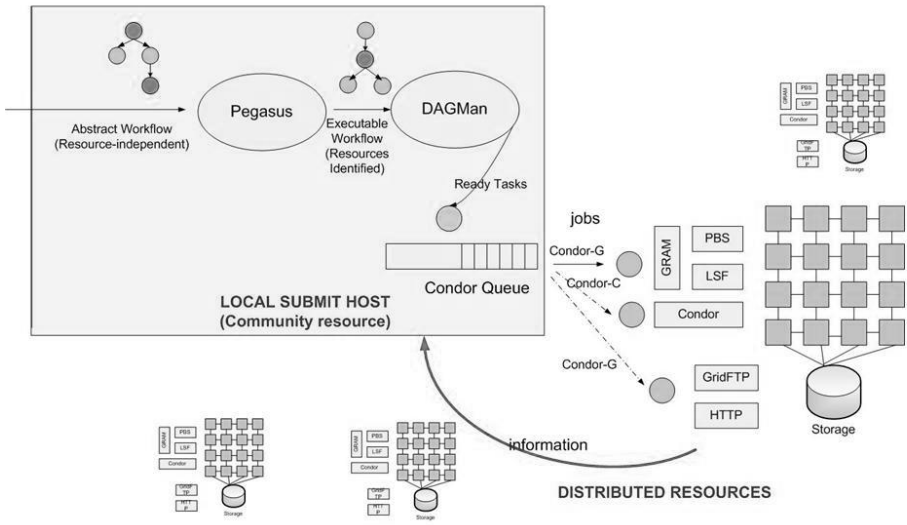


Figure 1: A General Overview of Pegasus and DAGMan Managing Workflows on the National Cyberinfrastructure.

The workflow mapping and workflow optimizations we perform as well as the scalability and robustness of our workflow execution enable us to efficiently and

reliably run large-scale scientific workflows. Below we describe some of these key features.

1.1. Automatically locating physical locations for both workflow components and data.

Mapping the workflow instance to an executable form involves finding the resources that are available and can perform the computations, the data that is used in the workflow, and the necessary software. We assume that data may be replicated in the environment and that users publish their data products into some data registry. This registry can be a private or community resource. Some communities, such as Laser Interferometer Gravitational-Wave Observatory (LIGO) [13] maintain project-wide registries of the data coming off the detectors. Pegasus uses the logical filenames referenced in the workflow to query a data registry service such as the Globus Replica Location Service (RLS) [21] to locate the replicas of the required data. Given the set of logical filenames, RLS returns a corresponding set of physical file locations. Optionally, Pegasus also adds nodes to the workflow to register the final and intermediate workflow data products into the registry. In this way, new data products can be easily discovered by the user, the community, or another workflow. In order to be able to find the location of the logical application component names (transformations) defined in the workflow instance, Pegasus queries the Transformation Catalog (TC) [25] and obtains the physical locations of the transformations (on possibly several systems) and the environment variables and libraries necessary for the proper execution of the software. Pegasus also supports staging of statically linked executables on demand. In that case, the executables are treated as input data for the corresponding workflow tasks. The executables are transferred to the remote grid sites along with other input data required by the jobs.

1.2. Finding appropriate resources to execute the components

Pegasus queries cyberinfrastructure monitoring services (the Globus Monitoring and Discovery Service (MDS) [24], the OSG VORS system [6], or any information service) to find the available resources and their characteristics (machine load, scheduler queue length, available disk space, and others). This information is combined with information from the Transformation Catalog to make scheduling decisions. When making a resource assignment, Pegasus prefers to schedule the computation where the data already exist; otherwise it makes a random choice or uses simple scheduling heuristics such as min-min [19], or HEFT [53]. Oftentimes it is difficult to apply sophisticated scheduling algorithms because of the incomplete or out-of-date information about resources, in-exact or missing models of the application performance, and the size of the workflows that may result in significant runtimes for the scheduling algorithms.

Pegasus also uses information services to find information about the location of the data movement services (GridFTP [10], RFT [11], or SRB [14] servers) that can perform wide-area data transfers, job managers [23] that can schedule jobs on the remote sites, storage locations, where data can be pre-staged, shared execution directories, site-wide environment variables, etc. This information is necessary to produce the executable workflow that describes the necessary data movement, computation and catalog updates. Registries of code and data as well as information

services allow Pegasus to provide a level of abstraction to the user and give it the freedom to automatically optimize the workflow execution.

1.3. Performance optimization through workflow restructuring

During the mapping process, Pegasus is able to restructure the workflow to improve the overall workflow performance. One of the optimizations involves clustering of workflow tasks so that they are treated as one for the purpose of submission to a remote site and where they are then expanded automatically into a sequence of tasks or a parallel set of tasks (via MPI [38]) for the purpose of execution.

Pegasus currently implements level- and label- based clustering. In level-based clustering, tasks at the same level can be clustered together. The user can specify either the number of clusters to be created per level or the number of tasks to be grouped in a cluster. We apply clustering techniques in applications such as Montage [5, 16, 17], an application, where there are often many (~10,000) short duration (order of seconds) tasks. This type of application usually incurs large overheads if each task is submitted individually. Figure 2 shows a small Montage workflow in its original form (left), clustered with two clusters per level (middle) and two tasks per cluster (right).

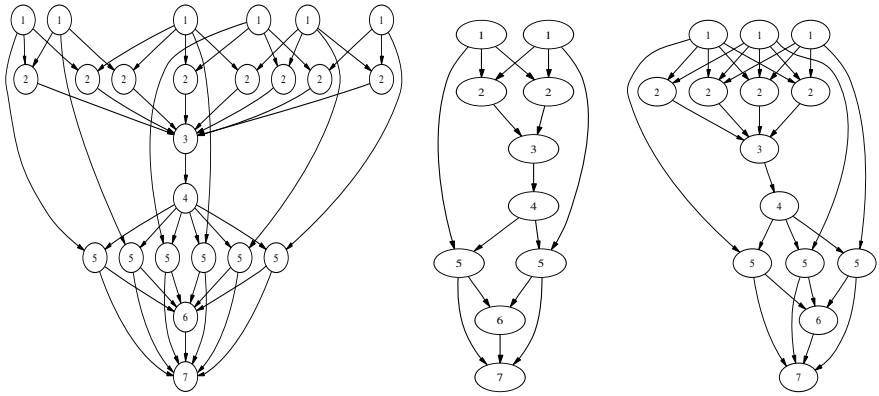


Figure 2. Montage workflow: original (left) clustered with two clusters per level (middle) and two tasks per cluster (right).

In label-based clustering, the user can label the tasks in the workflow to be clustered together. The tasks in the workflow with the same label are grouped into a single cluster. Figure 3(1) shows a workflow where tasks are labeled as *cluster_1* and *cluster_2* and the resulting clustered workflow is shown in Figure 3(2). Note that if the tasks in the clusters are executed sequentially implicit dependencies between tasks are added. Any clustering scheme can be implemented using an appropriate labeling scheme.

Each cluster whether generated using level- or label- based clustering must satisfy the convexity requirement that dictates that all paths between any two tasks in a cluster must be completely contained within it. The cluster shown in Figure 4 is non-convex since the path from t1 to t3 through t4 is not contained within the cluster. The difficulty here is that t4 must start execution after t1 has completed and before t3 starts execution.

Thus it creates co-scheduling requirements between clusters. However, due to the best effort nature of the execution environment, it is not possible to achieve co-scheduling without explicit resource control.

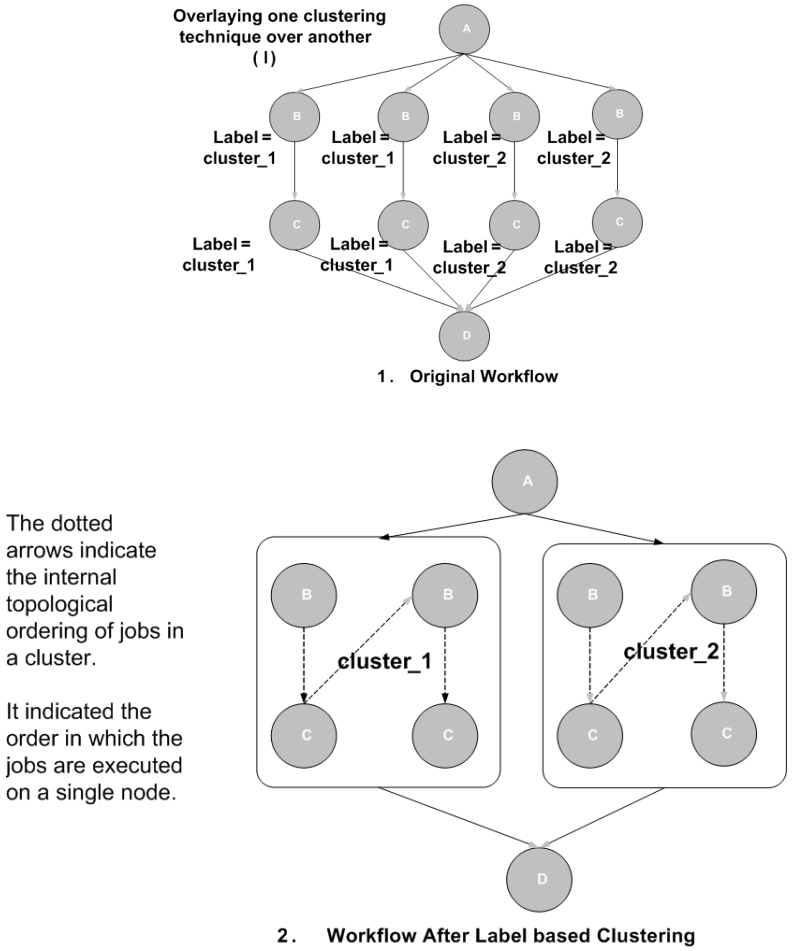


Figure 3. Example of label based clustering.

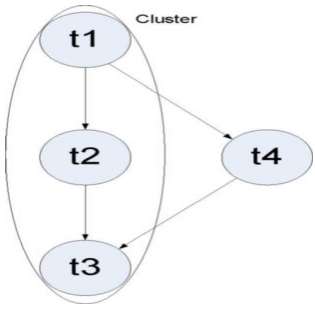


Figure 4. A non-convex cluster.

Pegasus does error checking to ensure that each cluster created by grouping the tasks with the same label satisfies the convexity requirement. Note that the clusters generated using level-based clustering trivially satisfy the convexity requirement since all the tasks at a level are independent of each other and no path exists between them. Another restriction of clustering is that the tasks within a cluster be scheduled to the same resource.

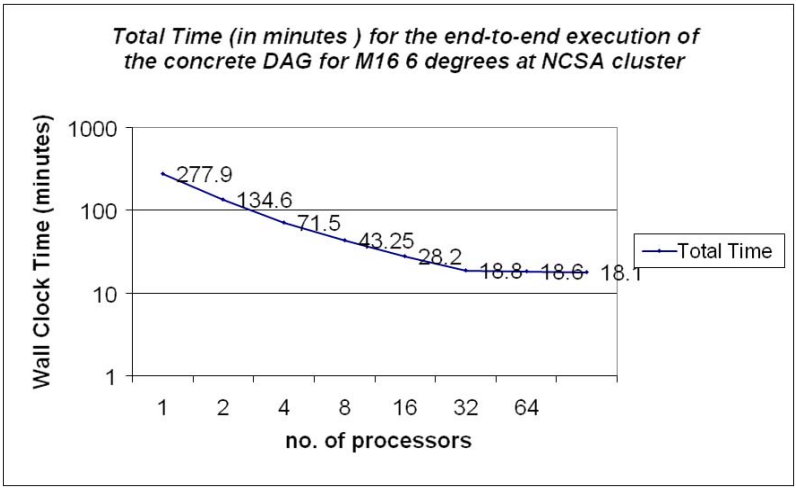


Figure 5: Improving Workflow Scalability With the Use of Clustering Techniques [40].

Figure 5 shows the results of the application of this optimization to Montage. The figure shows the results when Pegasus groups the nodes of the workflow into as many tasks per cluster as there are processors. Pegasus then schedules the clusters through DAGMan. Each cluster is executed as an MPI job on the TeraGrid. The figure shows a good speedup of approximately 15 on 32 processors. As the number of processors increases, the speedup decreases due to sequential aspects of the Montage workflow structure.

1.4. Workflow footprint optimization

In many execution environments, data storage available for executing applications can be limited. In our recent work [48, 51] we are studying mechanisms for minimizing the amount of disk space a particular workflow requires by adding nodes to the workflow to explicitly remove the data when they are no longer needed. The purpose of the cleanup job is to delete the data file from a specified computational resource to make room for the subsequent computations. Since a data file can be potentially replicated on multiple resources (in case the compute tasks are mapped to multiple resources) the decisions to add cleanup jobs are made on a per-resource basis. The algorithm is applied after the executable workflow has been created but before the workflow is executed. Details of the algorithm are described in [51], here we just provide a simple illustration.

Figure 6 shows an executable workflow containing 7 compute jobs {0,1,..,6} mapped to two resources {0,1}. The algorithm first creates a subgraph of the executable workflow for each execution resource used in the workflow. The subgraph of the workflow on resource 0 contains jobs {0,1,3,4} and the subgraph on resource 1 contains jobs {2,5,6}. The cleanup nodes added to this workflow using our algorithm are shown in Figure 7. The cleanup job for removing file f on resource r is denoted as C_{fr}.

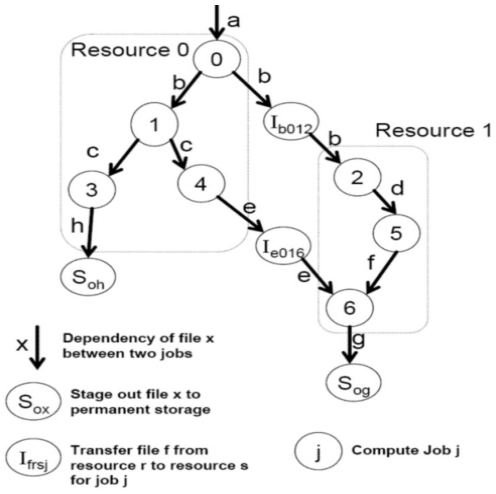


Figure 6: The Original Executable Workflow Mapped by Pegasus to Two resources.

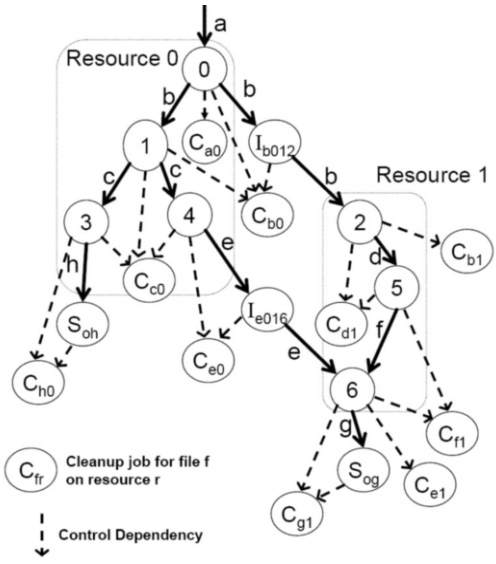


Figure 7**:** The Workflow with Dynamic Cleanup Nodes Added.

For each task in the subgraph, a list of files either required or produced by the task is constructed. For example, the list of files for task 1 mapped to resource 0 contains files b and c. For each file in the list, a cleanup job for that file on that resource is

created (if it does not already exist) and the task is made the parent of the cleanup job. Thus, a cleanup job, C_{c0}, which will remove file c on resource 0 is created and task 1 is made the parent of this cleanup job. The cleanup jobs for some files might already have been created as a result of parsing previous tasks. For example, the cleanup job C_{b0} for removing file b on resource 0 already exists (as a result of parsing task 0). In this case the task being parsed is added as a parent of the cleanup job. Thus, task 1 is added as a parent of cleanup job C_{b0}. When the entire subgraph has been traversed, there exists one cleanup job for every file required or produced by tasks mapped to the resource. If a file required by a task is being staged-in from another resource, then the algorithm makes the cleanup job for the file on the source resource a child of the stage-in job, ensuring that the file is not cleaned up on the source resource before it is transferred to the target resource. For example, file b required by task 2 mapped to resource 1 is being staged-in from resource 0 using stage-in job I_{b012}, and so the cleanup job for file b on resource 0 (C_{b0}) is made a child of I_{b012}. Finally, if a file produced by a task is being staged-out to a storage location, the cleanup job is made a child of the stage-out job. For instance, the cleanup job C_{h0} for removing file h on resource 0 is made a child of the stage-out job S_{oh} that stages out file h to permanent storage. By adding the appropriate dependencies, the algorithm makes sure that the file is cleaned up only when it is no longer required by any task in the workflow. In this version of the algorithm there are as many cleanup jobs as there are files. For large scientific workflows, this solution is not scalable, so for the final version of the algorithm, we reduced the number of clean up jobs to be at most as many as there are computational tasks in the workflow.

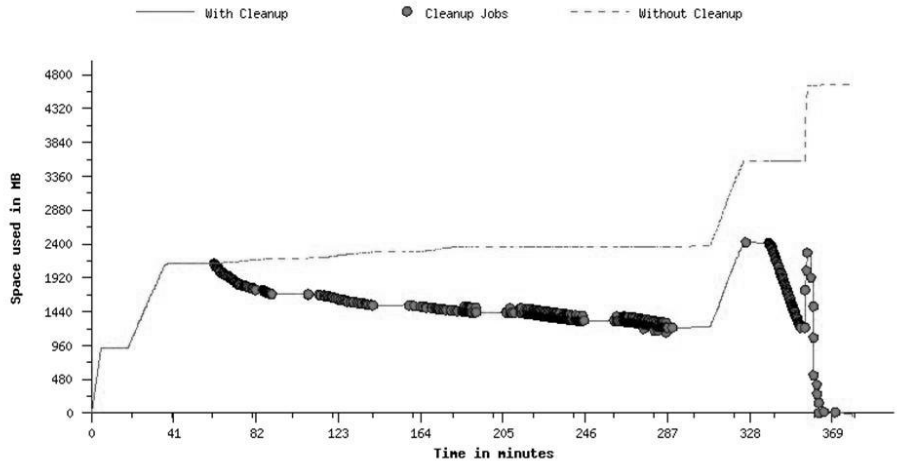


Figure 8: Actual Data Space Optimization for a Montage Workflow. The figure shows the amount of space used by a workflow over time on resources of Open Science Grid with and without space optimization[51].

In real experiments using the resources of the Open Science Grid, we were able to reduce the workflow data footprint of a 2 square degree Montage workflow by approximately 48% [51]. Figure 8 shows the storage space used by the workflow without and with cleanup during the execution of the workflow. Cleanup opportunities are widespread in the workflow as can be seen by the execution of cleanup jobs in the figure leading to a significant reduction in maximum storage used. For some workflows such as LIGO these opportunities might be concentrated towards the end of the

workflow execution resulting in relatively small savings in the storage space. For these workflows we use restructuring techniques in order to minimize the amount of storage space used. We are currently evaluating those techniques using real applications and cyberinfrastructure deployments.

1.5. Adapting to the changing execution environment

Very large workflows (on the order of thousands of tasks) should not be mapped all at once as the state and the availability resources can change significantly over the execution time of the workflow. Rather, portions of the workflow need to mapped and executed before the remaining portions are mapped and executed. In our work we have developed a workflow partitioning technique where the workflow is divided into smaller sub-workflows [26]. The partitioning algorithm is a pluggable component of Pegasus. The only restriction is that the partition preserves the dependencies in the original workflow and that there cannot be any cyclic dependencies between the partitions, or in other words, the partitions need to be convex as defined in the clustering section. Currently, we have implemented three basic partitioning algorithms: level-based, which creates partitions based on the depth of the workflow—all tasks at the same depth are in the same partition; single-node where each node is its own partition, and label-based partitioning. Single node partitioning is equivalent to just-in-time mapping, where each task is mapped to a resource only when it is ready to run. In label-based partitioning, the partitions are built based on the labels assigned to them allowing for a flexible partition structure. Figure 9 shows how a label-based partitioning works. Figure 9 (a) shows the original workflow labeled for partitioning. Figure 9 (b) shows this workflow partitioning into three partitions. In this workflow Partition 1 will execute first, then the other two partitions can execute in parallel. We also notice that although this partitioning is correct as it does not violate any dependencies, it does introduce additional dependencies, in particular between tasks C and D and task B. This may result in performance degradation.

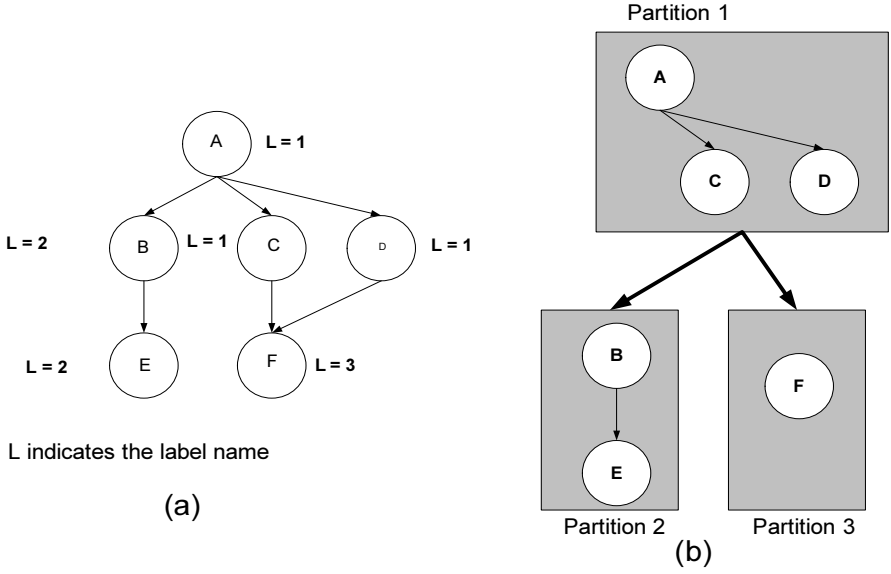


Figure 9: (a) A Workflow Labeled for Partitioning. (b) Resulting workflow.

Each partition is mapped by Pegasus and executed by DAGMan before the dependent partitions are treated in the same fashion. The entire process of managing the mapping and execution of workflows and following the dependencies between them is performed by DAGMan. The workflow partitioning approach combined with node clustering is often used by LIGO workflows running on the OSG and in SCEC workflows running on the TeraGrid to adjust to the dynamic grid conditions and to improve the overall workflow performance.

1.6. Support for reliable execution across a variety of platforms

Pegasus' partitioning and data reuse play an important role in the reliability of the workflow execution, especially in very dynamic execution environments. Breaking up the workflow into smaller pieces and mapping and executing at that level of granularity enables us to re-plan and re-execute the individual partitions when failures during execution occur [28]. If a partition fails during the mapping or the execution, we can trigger a retry (or several retries) of that partition. Additionally when the re-planning occurs within the partition, not everything needs to be re-executed because of the data-reuse capabilities of Pegasus. If for example the original sub-workflow consisted of three sequential tasks *t1, t2, t3*, each producing file *f1, f2, f3* respectively and task *t3* failed, then upon re-planning, Pegasus would discover that files *f1* and *f2* already exist and thus tasks *t1* and *t2* do not need to be re-executed and the only task that will be re-mapped and re-executed is task *t3*.

Additional reliability is provided by the DAGMan execution engine. DAGMan sits as a layer "above" the batch system in the software stack. DAGMan utilizes the batch system's standard API and logs in order to submit, query, and manipulate jobs, and does not directly interact with the jobs independently. While DAGMan currently only works with Condor [43] as a batch system, it can uses Condor's grid abilities (known as Condor-G) to submit to many other batch and grid systems. DAGMan reads the logs of the underlying batch system to follow the status of submitted jobs rather than invoking interactive tools or service APIs. Reliance on simpler, file-based I/O allows DAGMan's own implementation to be simpler, more scalable and reliable across many platforms, and therefore more robust. For example, if DAGMan has crashed while the underlying batch system continues to run jobs, DAGMan can recover its state upon restart (by reading logs provided by the batch system) and there is no concern about missing callbacks or gathering information if the batch system is temporarily unavailable ─ it is all in the log file.

Workflow management includes not only job submission and monitoring but job preparation, cleanup, throttling, retry, and other actions necessary to ensure the successful workflow execution. DAGMan attempts to overcome or work around as many execution errors as possible, and in the face of errors it cannot overcome, it allows the user to resolve the problem manually and then resume the workflow from the point where it last left off. This can be thought of as a "checkpointing" of the workflow, just as some batch systems provide checkpointing of jobs.

1.7. Scalability of Pegasus and DAGMan

The scalability of our workflow management system can be seen in the context of scientific applications that use our technologies.

The Southern California Earthquake Center (SCEC) [8] uses our tools to produce more accurate seismic hazard maps. These maps, generated as part of the CyberShake project [45], indicate the maximum amount of shaking expected at a particular geographic location over a certain period of time. The hazard maps are used by civil engineers to determine building design tolerances. Figure 10 shows the results of one of two major CyberShake executions. The run was performed on the TeraGrid in the fall of 2005 and used Pegasus and DAGMan. The two workflows ran over a period of 23 days and processed 20TB of data using 1.8 CPU Years. The total number of tasks in the two workflows was 261,823. CyberShake workflows have tasks with very varied runtimes as can be seen in Figure 11. The task times range from l minute to almost 70 hours.

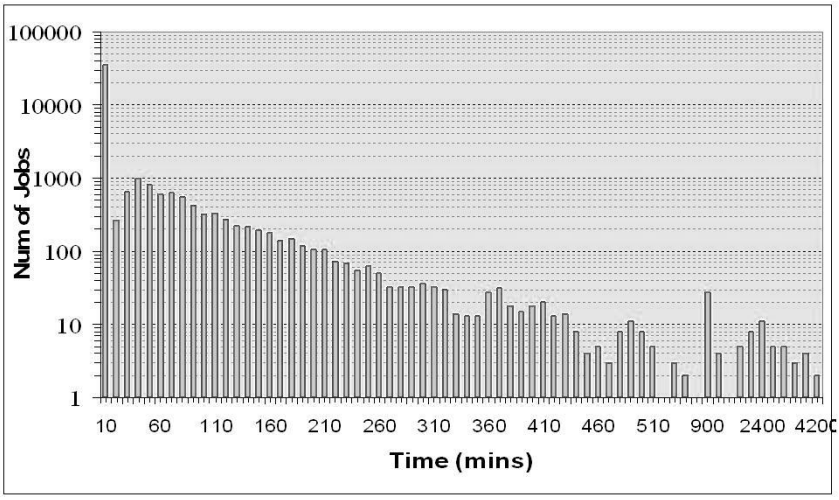


Figure 10: Execution of SCEC Workflows on the TeraGrid in 2005 [29, 30].

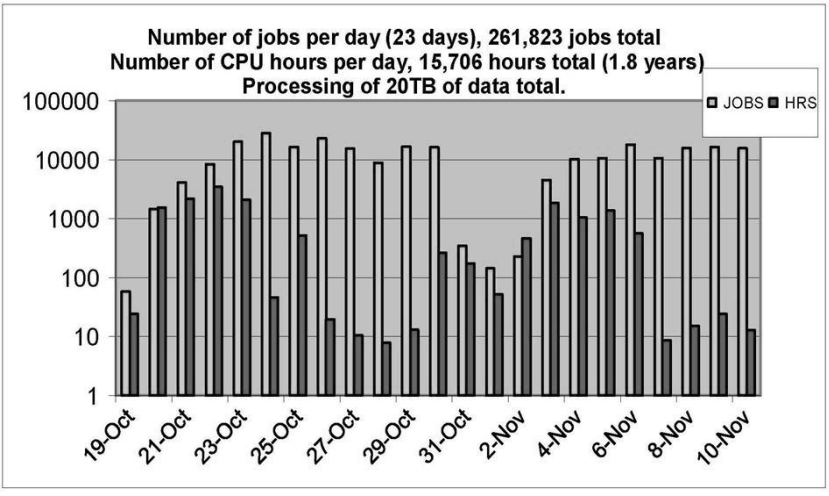


Figure 11: Distribution of Seismogram Tasks in the SCEC CyberShake Workflow.

After several months of validation, result interpretation, code modification and improvement, SCEC conducted 10 more runs of CyberShake in the Spring of 2006. The total number of tasks was over 212,000 and the runtime was approximately 8 CPU months. Since then the scientists have been analyzing the results and trying to understand why the results differ from the traditional calculations. Now, a new wave of simulations is underway. As the result of many simulations, SCEC scientists now believe that CyberShake may be more accurate than traditional methods because it models rupture directivity and sedimentary basin effects which contribute to the shaking experienced at different geographic locations. As a result more accurate hazard maps can be created. SCEC is also using Pegasus and DAGMan in the Earthworks Portal [46], a TeraGrid Science Gateway, hosted at Washington University that allows users to configure and execute earthquake wave propagation simulations structured as workflows through a simple portal interface.

Pegasus and DAGMan are used in the LIGO project to map binary inspiral analysis workflows onto the OSG [20]. A month of LIGO data requires many thousands of jobs, running for days on hundreds of CPUs. Figure 12 illustrates the use of OSG for the LIGO workflows over the period of November 2006 to early January 2007. The figure was created using the Monalisa monitoring software used on OSG [9]. The workflows were run across several OSG sites and used a total of 2.5 CPU years of computing over a period of two months.

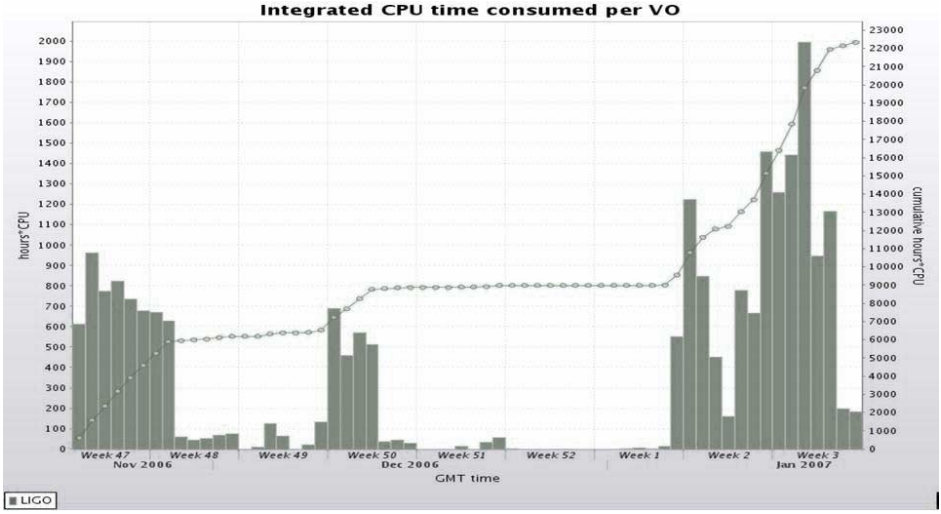


Figure 12: LIGO's CPU Hours Usage of OSG Resources.

Currently DAGMan is also used by a broad range of application in both scientific and commercial domains. Some examples of DAGMan use in science are in the areas of bioinformatics, machine translation, computational fluid dynamics, high-energy physics, and others. In particular DAGMan is used in production by BioMagResBank (BMRB) [3] at UW-Madison to execute BLAST workflows, which consist of approximately 500 nodes and run for approximately 24 hours on 100 processors. During a workflow run, the BMRB's BLAST setup compares 4,000 sequences against multiple databases containing a total of ~4.5 million entries. Another use of DAGMan is the NMI build and test infrastructure [4]. This infrastructure automatically builds a

set of software packages every day on a variety of platforms and performs automated tests on the resulting builds. The builds are represented as DAGMan workflows, where the dependencies reflect the dependencies between the builds of the software packages. Figures 13 and 14 show some of the details of the execution of the workflows over 28 months starting in October of 2004. Over that time period there were ~2,800 workflows submitted each month, with a total of over 105,000 workflow tasks executed each month.

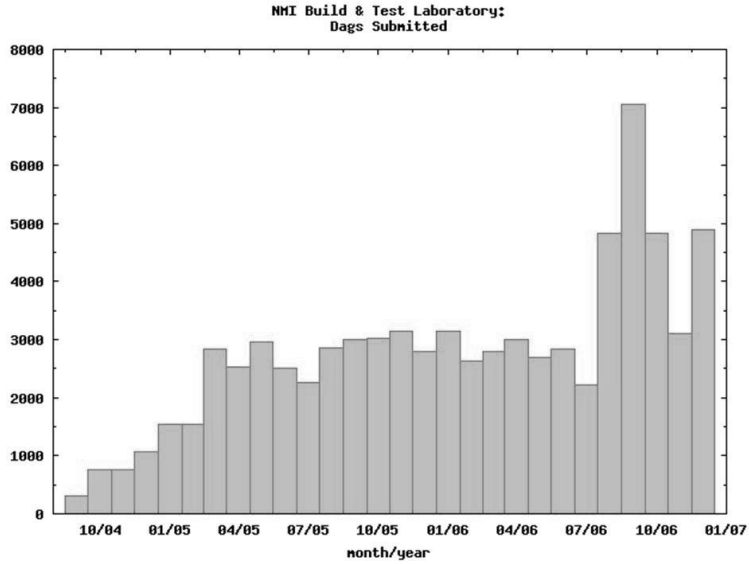


Figure 13: The Number of Workflow Submitted Each Month in Support of the NMI Build and Test.

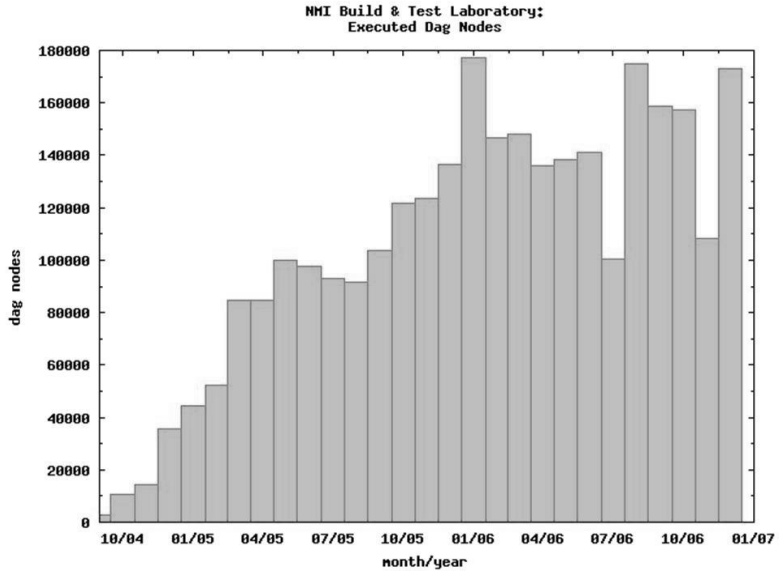


Figure 14: The Number of Workflow Tasks Submitted Each Month in Support of the NMI Build and Test.

1.8. Provenance Tracking

One promising area in large-scale applications in general and in workflow technologies in particular is the generation, management and querying of provenance which provides information about how data was produced. In our current system, Pegasus stores provenance information related to the execution of the workflow components. This information describes the execution environment of the tasks, where they were executed, how long the execution took place, which files were used and generated and other task-level information. We are also exploring the use of provenance tracking capabilities to record the workflow transformations performed by Pegasus during the workflow mapping process [41].

2. Other Applications Using Pegasus and DAGMan

Pegasus and DAGMan are used in astronomy, and in particular in the Montage application which delivers science-grade mosaics of the sky. Our workflow technologies were used to transform a single-processor Montage code into a complex workflow and parallelized computations to process larger-scale images. Montage workflows mapped by Pegasus to the existing cyberinfrastructure are characterized by tens of thousands of executable tasks and the processing of thousands of images. Recently, Montage was used to make a scientific discovery: the verification of a bar in the spiral galaxy M31 [15]. Although there have been hints of a bar in M31 from optical data, none of the analyses were convincing because the effects of interstellar extinction at optical wavelengths were severe. However, the universe is much more transparent in the infrared, and this enabled astronomers to overcome the effects of interstellar extinction. There was one more problem: the variable background in the infrared images hid the structure of the galaxy. By using Montage, which was able to rectify the backgrounds to a common level, the astronomers were finally able to see the structure.

Pegasus and DAGMan are also used in the Telescience project [42] and portal to support 3D reconstruction of electron tomography images. The UCSD scientists plan to continue to rely on our workflow technologies to expand the set of Grid applications they support within their portal environment and to develop new techniques that can provide real-time feedback from the 3D reconstruction to the scientists manipulating the instrument. Data mining and natural language processing applications are new user communities that are exploring the use of our workflow technologies to manage the large-scale computations on today's cyberinfrastructure.

Within GriPhyN [12], Pegasus was made available as part of the Virtual Data System (VDS) and interfaced with VDL. Today several applications use Pegasus and DAGMan it in that fashion. Among them are: a climate modeling application, where simulations which used to take 2.5 months to run manually, took only 2.5 days to run using our tools [47]. Another example is the GADU project (Genome Analysis and Database Update) [49], where the researchers at Argonne National Laboratory use Pegasus and DAGMan to conduct genome analysis on OSG.

3. Conclusions

We have shown the ability of the end-to-end Pegasus/DAGMan to efficiently and robustly execute computational workflows in a variety of application domains. Although we were able to address many issues faced by today's large-scale analyses running on the distributed cyberinfrastructure resources, many challenges still remain.

Through our work with data-intensive computations, we have recognized that data management is at least just as important as computation scheduling. When data sets being processed or generated by workflows are significant—sometimes on the order of Terabytes, it is critical to employ workflow scheduling techniques that are aware of the storage necessary to support the successful workflow execution.

Since many scientific collaborations are managing their experimental data across the distributed environment, it is also necessary for the workflows to be aware of these systems and collaborate with them in order to support the overall analysis. Examining these data-versus computation-placement issues that are being implemented by independent systems remains a challenge and we plan to continue our work in this area [27].

Acknowledgments

This work was supported by the National Science Foundation under grant CNS 0615412 and EAR-0122464. This research was done using resources provided by the Open Science Grid, which is supported by the National Science Foundation and the U.S. Department of Energy's Office of Science and provided by the NSF TeraGrid.

The authors would like to thank the many collaborators from the science projects, among them: Tom Jordan, Phil Maechling, Robert Graves, David Meyers, and Scott Callaghan (SCEC), Kent Blackburn, Duncan Brown, Scott Koranda, and Britta Daudert (LIGO), Bruce Berriman, John Good, and Dan Katz (Montage), Steven Peltier and Abel Lin (UCSD-NCMIR), Yolanda Gil, Jihie Kim and Varun Ratnakar (Wings), Luc Moreau, Simon Miles, and Paul Groth (PASOA provenance), Rizos Sakellariou (workflow scheduling), and Jens Voeckler (remote execution).

References

[1] "TeraGrid."http://www.teragrid.org/
[2] "Open Science Grid."www.opensciencegrid.org
[3] "Biological Magnetic Resonance Data Bank," http://www.bmrb.wisc.edu/
[4] "NMI Build and Test Lab."http://nmi.cs.wisc.edu/
[5] "Montage."http://montage.ipac.caltech.edu
[6] "VO resource selector," 2006.http://vors.grid.iu.edu/cgi-bin/index.cgi
[7] *Workflows in e-Science*. I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.: Springer, 2006.
[8] "Southern California Earthquake Center (SCEC)," 2006.http://www.scec.org/
[9] "OSG Monitoring System Monalisa," 2007.http://grid02.uits.indiana.edu:8080/
[10] W. Allcock, et al., "Data Management and Transfer in High-Performance Computational Grid Environments," *Parallel Computing,* 2001.

[11] W. E. Allcock, et al., "Reliable Data Transport: A Critical Service for the Grid," in *Building Service Based Grids Workshop, Global Grid Forum 11*, 2004.
[12] P. Avery and I. Foster, "The GriPhyN Project: Towards Petascale Virtual Data Grids," Technical Report, GriPhyN-2001-15, 2001. www.griphyn.org
[13] B. C. Barish and R. Weiss, "LIGO and the Detection of Gravitational Waves," *Physics Today,* vol. 52, p. 44, 1999.
[14] C. Baru, et al., "The SDSC Storage Resource Broker," in *Proc. CASCON'98 Conference*, 1998.
[15] R. L. Beaton, et al., "Unveiling the Boxy Bulge and Bar of the Andromeda Spiral Galaxy," *Astrophysical Journal Letters* 2006 (in press).
[16] B. Berriman, et al., "Montage: A Grid-Enabled Image Mosaic Service for the NVO," in *Astronomical Data Analysis Software & Systems (ADASS) XIII*, 2003.
[17] G. B. Berriman, et al., "Montage: A Grid Enabled Engine for Delivering Custom Science-Grade Mosaics On Demand," in *SPIE Conference 5487: Astronomical Telescopes*, 2004.
[18] G. B. Berriman, et al., "Generating Complex Astronomy Workflows," in *Workflows in e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.: Springer, 2006.
[19] J. Blythe, et al., "Task Scheduling Strategies for Workflow-based Applications in Grids," in *CCGrid*, Cardiff, UK, 2005.
[20] D. A. Brown, et al., "A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis," in *Workflows for e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.: Springer, 2006.
[21] A. Chervenak, et al., "Giggle: A Framework for Constructing Sclable Replica Location Services.," in *Proceedings of Supercomputing 2002 (SC2002)*, 2002.
[22] P. Couvares, et al., "Workflow Management in Condor," in *Workflows in e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.: Springer, 2006, pp. 357-375.
[23] K. Czajkowski, et al., "A Resource Management Architecture for Metacomputing Systems," in *4th Workshop on Job Scheduling Strategies for Parallel Processing*: Springer-Verlag, 1998, pp. 62-82.
[24] K. Czajkowski, et al., "Grid Information Services for Distributed Resource Sharing," in *10th IEEE International Symposium on High Performance Distributed Computing*, 2001, pp. 181-184.
[25] E. Deelman, et al., "Transformation Catalog Design for GriPhyN," Technical Report, GriPhyN-2001-17, 2001. www.griphyn.org
[26] E. Deelman, et al., "Pegasus : Mapping Scientific Workflows onto the Grid," in *2nd EUROPEAN ACROSS GRIDS CONFERENCE*, Nicosia, Cyprus, 2004.
[27] E. Deelman, et al., "What Makes Workflows Work in an Opportunistic Environment?," *Concurrency and Computation: Practice and Experience,* vol. 18, 2005.
[28] E. Deelman, et al., "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems," *Scientific Programming Journal,* vol. 13, pp. 219-237, 2005.
[29] E. Deelman, et al., "Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The CyberShake Example,"

E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, p. 14, 2006.

[30] E. Deelman, et al., "Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance tracking: The CyberShake Example," in *e-Science*, Amsterdam, The Netherlands, 2006.

[31] E. Deelman, et al., "Pegasus: Mapping Large-Scale Workflows to Distributed Resources," in *Workflows in e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.: Springer, 2006.

[32] D. H. J. Epema, et al., "A Worldwide Flock of Condors: Load Sharing among Workstation Clusters," *Future Generation Computer Systems,* vol. 12, 1996.

[33] I. Foster, et al., "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," in *Scientific and Statistical Database Management*, 2002.

[34] J. Frey, et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids," *Proc. of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10),* 2001.

[35] J. Frey, et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids.," *Cluster Computing,* vol. 5, pp. 237-246, 2002.

[36] Y. Gil, et al., "Wings for Pegasus: A Semantic Approach to Creating Very Large Scientific Workflows," in *OWL: Experiences and Directions (OWL-ED)*, Athens, GA, 2006.

[37] Y. Gil, et al., "Wings for Pegasus: Creating Large-Scale Scientific Applications Using Semantic Representations of Computational Workflows," in *Proceedings of the 19th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI)* Vancouver, British Columbia, Canada, 2007 (to appear)

[38] W. Gropp, et al., *Using MPI: Portable Parallel Programming with the Message Passing Interface*: MIT Press, 1994.

[39] R. Henderson and D. Tweten, "Portable Batch System: External Reference Specification," 1996.

[40] D. S. Katz, et al., "Comparison of Two Methods for Building Astronomical Image Mosaics on a Grid," in *International Conference on Parallel Processing Workshops (ICPPW'05)*, 2005.

[41] J. Kim, et al., "Provenance Trails in the Wings/Pegasus System," *Concurrency and Computation: Practice and Experience,* 2006 (in submission).

[42] A. Lathers, et al., "Enabling Parallel Scientific Applications with Workflow Tools," in *Challenges of Large Applications in Distributed Environments (CLADE)*, Paris, 2006.

[43] M. Litzkow, et al., "Condor - A Hunter of Idle Workstations," in *Proc. 8th Intl Conf. on Distributed Computing Systems*, 1988, pp. 104-111.

[44] B. Ludäscher, et al., "Scientific Workflow Management and the Kepler System," *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows,* 2005.

[45] P. Maechling, et al., "SCEC CyberShake Workflows---Automating Probabilistic Seismic Hazard Analysis Calculations," in *Workflows for e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.: Springer, 2006.

[46] J. Muench, et al., "SCEC Earthworks Science Gateway: Widening SCEC Community Access to the TeraGrid," TeraGrid 2006 Conference, 2006

[47] V. Nefedova, et al., "Automating Climate Science: Large Ensemble Simulations on the TeraGrid with the GriPhyN Virtual Data System," in *e-Science*, Amsterdam, The Netherlands, 2006.

[48] A. Ramakrishnan, et al., "Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources," in *Seventh IEEE International Symposium on Cluster Computing and the Grid - CCGrid 2007*, 2007 *(to appear)*

[49] A. Rodriguez, et al., "A Grid-enabled service for high-throughput genome analysis," in *Global Grid Forum 10, Workshop on Case Studies on GridApplications* Berlin, 2004

[50] G. Singh, et al., "The Pegasus Portal: Web Based Grid Computing," in *20th Annual ACM Symposium on Applied Computing, SAC*, 2005.

[51] G. Singh, et al., "Optimizing Workflow Data Footprint," *Scientific Programming Journal, Special issue on Dynamic Computational Workflows: Discovery, Optimization, and Scheduling,* 2007 (to appear).

[52] I. Taylor, et al., "Distributed P2P Computing within Triana: A Galaxy Visualization Test Case.," in *IPDPS 2003*, 2003.

[53] H. Topcuouglu, et al., "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing " *J IEEE Trans. Parallel Distrib. Syst. ,* vol. 13, pp. 260-274 2002

[54] V. Welch, et al., "Security for Grid Services.," in *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*, 2003.

[55] S. Zhou, "LSF: Load Sharing in Large-Scale Heterogeneous Distributed Systems," in *Proc.\ Workshop on Cluster Computing*, 1992.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Building an Infrastructure for Urgent Computing

Pete BECKMAN [a] Ivan BESCHASTNIKH [b] Suman NADELLA [c] and Nick TREBON [c]

[a] *Mathematics and Computer Science Division, Argonne National Laboratory*
9700 S. Cass Avenue, Argonne, IL 60439
[b] *Computer Science and Engineering, University of Washington*
Box 352350, Seattle, WA 98195
[c] *Computation Institute, The University of Chicago*
5801 S. Ellis Avenue, Chicago, IL 60637

Abstract. Large-scale scientific computing is playing an ever-increasing role in critical decision-making and dynamic, event-driven systems. While some computation can simply wait in a job queue until resources become available, key decisions concerning life-threatening situations must be made quickly. A computation to predict the flow of airborne contaminants from a ruptured railcar must guide evacuation rapidly, before the results are meaningless. Although not as urgent, on-demand computing is often required to leverage a scientific opportunity. For example, a burst of data from seismometers could trigger urgent computation that then redirects instruments to focus data collection on specific regions, before the opportunity is lost. This paper describes the challenges of building an infrastructure to support urgent computing. We focus on three areas: the needs and requirements of an urgent computing system, a prototype urgent computing system called SPRUCE currently deployed on the TeraGrid, and future technologies and directions for urgent computing.

Keywords. Urgent Computing, On-Demand, Emergency, Disaster Response

Introduction

Since the days of the first supercomputers, computation has been playing an ever-increasing role in science. While the earliest supercomputers were used primarily to increase the speed of basic calculations that could otherwise be performed by hand, modern scientific simulation and modeling programs are extremely sophisticated. Often, large-scale codes represent decades of programmer effort. As the computer models have incorporated better computational methods and improved physics and chemistry, they have become more accurate. Their ability to accurately model and predict complex systems has led to advances in areas ranging from weather prediction to better traffic planning. For example, the latest report from the Intergovernmental Panel on Climate Change [1] relies heavily on computer models to peer into the future and explore what the Earth will be like as warming continues. While decision makers would like simulation results as

soon as possible, there is often little urgency or a deadline to complete the computation. Developing public policy is rarely fast.

However, there is a growing number of problem domains where key decisions must be made quickly with the aid of large-scale computation. In these domains, "urgent computing" is essential, and late results are useless. A computer model capable of determining where tornadoes will form must provide early warning to local residents. A computation to predict coastline flooding or avalanche danger must guide evacuation while there is still time. Furthermore, on-demand computing is often required to take advantage of a scientific opportunity, for example, to process data and steer activities during an experiment or observation of an unpredictable natural event. Without immediate access to large computational resources, the opportunity may be lost.

These on-demand, large-scale computations cannot wait endlessly in a job queue for Grid resources to become available. However, neither can the scientific community afford to keep multimillion dollar infrastructures idle until needed by an urgent computation. Instead, we must develop technologies that can support urgent computation dynamically and yet preserve overall machine utilization and the productivity of the scientists working daily on the systems.

1. The Needs of Urgent Computing

Urgent computing stems from dynamic work-flows and deadline-driven activities. Some computation needs more interactivity and immediate attention, while other computation is more time tolerant and suitable for traditional batch computing environments. This notion is not at all new; and in our daily computing, we have a rich range of models. Many new Web-based applications, from instant messengers to wikis, require users to be connected to the Internet. These applications are tolerant to network delays, however, and response delays can be several seconds before the user experience is significantly impacted. Even more tolerant are mail programs and source code repositories that queue transactions for later. At the other end of the spectrum are applications where users need near-instant feedback, such as voice-over-IP, remotely editing documents, or using a mouse to control the rendering and display of a large parallel data-set. Unfortunately, this rich continuum of responsiveness that we have come to expect from our Internet-based computing is not supported at most high-performance computing (HPC) centers. For most HPC systems, jobs are submitted to a batch queue where they will sit for some unknown length of time, or the user can request a reservation. Reservations fix the start time for the job; but because the scheduler uses worst-case execution times to guarantee the reservation, start times are pessimistic and often worse than what a user would experience by simply submitting to the batch queue.

Urgent computation is event-driven and deadline-based. To build an international infrastructure to support such computation, however, we must address more than the technical challenges. While questions such as "Which applications have execution deadlines not met by standard batch queues?" and "Under what circumstances should executing jobs be preempted?" are difficult to answer, we must begin to build a framework for discussing these topics and formulating fair policies to support the diverse user community.

2. A Framework for Urgent Computing

An urgent computing system requires more than a mechanism for elevating job priority. Some existing queueing systems and schedulers permit users to begin jobs sooner by using extra accounting units or by contacting an administrator for the computing resource. But effective urgent computing must be supported within a larger framework that supports interactions with user communities, applications, job queues, and decision makers. We believe it must also specifically address five important concepts: urgent computing session, priority policies, participation flexibility, allocations, and verification drills.

2.1. Session Activition

We are all familiar with on-demand, urgent situations at work and in our community. From the warning light on an automobile dashboard to an automated phone pager alert that a supercomputer is down, urgent situations begin with a trigger. Based on operating protocols, the event may initiate a response, or "session." Once initiated, the session is not concluded until someone evaluates the circumstances and determines that special actions are no longer required. Likewise, urgent computing jobs must occur within a clearly defined session. For some applications, hundreds or thousands of jobs might be submitted and run during a session. For others, a workflow requiring several machines and instruments might be required. The details surrounding the activation of an urgent session must be carefully logged and recorded. In all cases, an audit trail that can be later scrutinized and used to improve policies and response times must be created. For urgent computing, system administrators must be notified when sessions begin, permitting periods of increased attentiveness and, if needed, human intervention to provide the resources required. When the session completes, everyone must also be notified that operations have returned to regular service.

2.2. Elevated Priority for Resources

A key component to urgent computing procedures is deadline and priority-based resource management. An ambulance with sirens blaring and lights flashing forces non critical vehicles to slow and yield the right-of-way. An urgent computing framework must permit important jobs to gain priority access to the CPU, disk, networking, and other key components required for simulations. Tasks and activities that slow or hamper the speedy completion of computer simulations must be avoided when time is critical. On some systems, deadline-based elevated priority may simply grant shorter wait times in the queue. On others, it may translate to immediate, dedicated access. Furthermore, simulations are always part of a workflow that includes setting up the problem, data acquisition, computation, and analysis of the results. Some applications may require the movement of massive quantities of data, where network bandwidth can become a limiting constraint. In such situations, elevated priority might be implemented with prioritization of data streams in the routers or with guaranteed bandwidth reservations.

2.3. Flexible Participation Policies

Unless deployed on a set of resources owned and managed by a single stakeholder, an urgent computing system must be able to accommodate many types of resources and

policies. The NASA Columbia supercomputer was designed to provide simulation and modeling support for active launch missions [2]. As such, operational policies and expectations can be designed from the top down. To build an international network of diverse resources to support urgent computation, however, participation policies must be flexible. Furthermore, the system must coexist with ongoing operations. For example, some HPC centers may support preemption for certain applications or priorities, while other HPC centers may provide only "next-to-run" priority following the normal completion of existing jobs. There may also be regional influences based on stake-holders. For example, California supercomputer centers may link all of their resources to support urgent modeling of infrastructure, such as roadways and gas lines, immediately following an earthquake, while supercomputer centers in the Rocky Mountains may work together to build a capability that can quickly predict where a wildfire might travel based on current weather data and fuel models of the terrain.

2.4. Allocation and Usage Policies

The first question that arises when discussing priority access for resources is "Who gets it?" Naturally, a key element to supporting on-demand and urgent computing is creating a clear set of policies for both allocation and activation. At the Urgent Computing Workshop held at Argonne National Laboratory in April 2007 [3], participants suggested that science teams needing urgent computing should quantify their requirements, from the frequency of need to the size and length of jobs, so HPC centers can better plan for their response. They should also explain their need, so requests can be prioritized. Similarly, even after an allocation for urgent computing is provided, clear activation policies must be formulated. A scientist given an urgent computing allocation for calculating flood levels should not use it to meet a paper deadline. Since urgent computing is a relatively new concept, clear policies for allocation and usage are needed to allay user concerns of favoritism or misuse.

2.5. Practice and Verification Drills

No system, regardless of its level of technical sophistication, can operate reliably without careful testing and verification. A system to support urgent computation must include support for testing and verification of components as part of the core infrastructure, not as an afterthought. In the same way that office buildings must periodically perform fire drills to test both the methods for alerting occupants as well as their training to evacuate quickly and calmly, an urgent computing framework must have periodic tests of the procedures, policies, and technology. We refer to urgent computing applications that have been periodically tested and their correctness verified as being in "warm standby." They are ready to run as soon as inputs are provided. Scientists must know both when the last successful test was performed and how often the application runs correctly and to completion on a particular platform. These values can help in the prediction and selection of computational resources.

3. SPRUCE

SPRUCE, the *S*pecial *PR*iority and *U*rgent *C*omputing *E*nvironment, is our implementation of the underlying components required to build an urgent computing framework.

SPRUCE uses a token-based authorization system chosen to facilitate allocation and tracking of urgent sessions. As a raw technology, there are no dictated policies within SPRUCE; resource providers have full control and flexibility to choose the policies they are comfortable with and implement them as they see fit. To build a complete solution for urgent computing, SPRUCE must be combined with allocation and activation policies, local participation policies for each resource, and procedures to support "warm standby" drills. These application drills not only verify end-to-end correctness, they also generate performance and reliability logs that can aid in resource selection.

3.1. Right-of-Way Tokens

Many possible authorization mechanisms could be used to let users initiate an urgent computing session, including digital certificates, signed files, proxy authentication, and shared-secret passwords. In time-critical situations, however, simpler is better. Relying on complex digital authentication and authorization schemes could easily become a stumbling block to quick response. Hence, simple transferable tokens were chosen for SPRUCE. This design is based on existing emergency response systems proven in the field, such as the priority telephone access system supported by the U.S. Government Emergency Telecommunications Service within the Department of Homeland Security [4]. Users of the priority telephone access system, such as officials at hospitals, fire departments, government offices, and 911 centers, carry a wallet-sized card with an authorization number. Cardholders can use the number to place high-priority phone calls that jump to the top of the queue for both land- and cell-based traffic even if circuits are completely jammed because of a disaster.

The SPRUCE tokens (see Figure 1) are unique 16-character strings that are issued to scientists who have permission to initiate an urgent computing session. The token need not be physical and could be hidden behind a Web portal or other middleware and used by automated systems. When a token is created, several important attributes are set.

- Resources that can be used for urgent jobs
- Maximum urgency that can be requested on any job
- Session lifetime (period for which urgent jobs may be submitted)
- Expiration date of the token
- Project name
- People to be notified if the token is used (e.g., local administrators)

A token represents a unique "session" that can include multiple jobs and lasts for a clearly defined period. A token can also be associated with a large group of collaborators; once the token is activated, anyone in the group can submit jobs with elevated priority. Users may be added or removed from the token at any time, providing flexible coordination. We emphasize that the right-of-way token is not related to machine access or authentication. Users must already have an account and be able to log on and authenticate in the traditional manner. The token allows the users only to begin and end an urgent computing session, during which they may request elevated priority for jobs. Without an active session, requests for elevated job priority are logged and ignored. Moreover, job priority is not elevated by default. Only jobs submitted with special, urgent job parameters, described in Section 3.4, receive unique treatment.

To support token-based session activation and access to elevated-priority resources, the SPRUCE architecture has three main components: user workflow and client-side job

Figure 1. SPRUCE "right-of-way" token

submission tools, the SPRUCE portal for token management and urgent request authentication, and local resource provider agents that respond to priority requests.

3.2. SPRUCE User Workflow

The SPRUCE workflow is designed for application teams that provide computer-aided decision support or instrument control. Each application team is organized by a principal investigator (PI). The PI selects the computational "first responders," senior staff who may initiate an urgent computing session by activating a token. First responders are responsible for evaluating the situation in light of the policies for using urgent computing. They must decide whether usage of the urgent resources is indeed warranted and meets the guidelines. The team must also have analysts or interpreters who can translate the raw data and simulation output into advice for decision makers or possibly feedback to instruments or controls. An application that models airflow across a city to evaluate contamination scenarios may have many subtle details that need interpretation and presentation to city managers as they formulate response scenarios.

Figure 2 illustrates how the SPRUCE workflow is initiated. The workflow begins as the result of a trigger, which may be automatic (e.g., an automated warning message from a weather advisory RSS feed parser) or human-generated (e.g., a phone call to the PI). SPRUCE token holders are expected to use tokens with discretion and according to coordinated policies, similar to the way that citizens are expected to use good judgment before dialing 911. Token usage will be monitored and reviewed. Administrators can revoke tokens at any time.

The first responder begins interaction with the SPRUCE system by initiating a session. Token activation can be done through a Web-based user portal or via a Web service interface. Systems built from the Web service interface can be automated and incorporated into domain-specific toolsets, avoiding human intervention. Activation is described in greater detail in Section 3.5.2. Often, running a large simulation involves numerous scientists who are responsible for tasks ranging from acquiring the most recent data-set to producing a visualization for analysis. The initiator of the SPRUCE session can in-

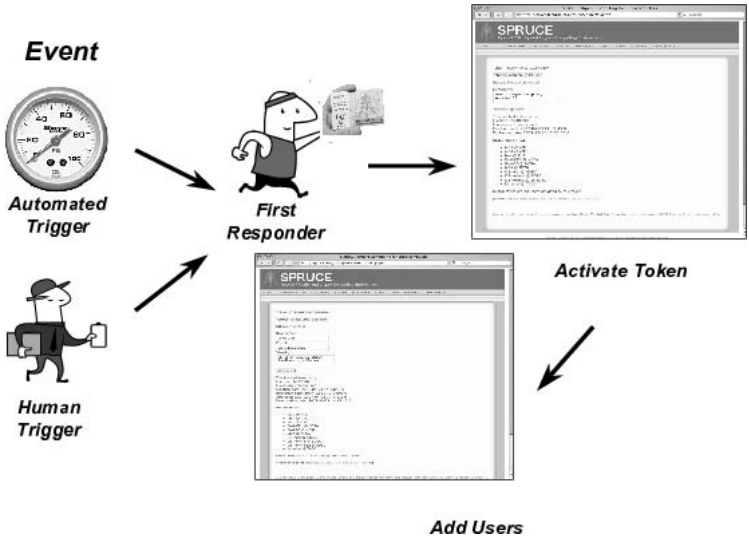


Figure 2. SPRUCE token activation

dicate which scientist or set of scientists will be able to request elevated priority while submitting urgent jobs. This set may later be augmented or edited.

3.3. Resource Selection

Once a token is activated and the application team has been specified, scientists can organize their computation and submit jobs. Naturally, there is no time to port the application to new platforms or architectures or to try a new compiler. Applications must be prepared for immediate use—they must be in "warm standby." All of the application development, testing, and tuning must be complete prior to freezing the code and marking it ready for urgent computation. Grids such as the TeraGrid have dozens of large-scale computational resources. The SPRUCE architecture supports large, diverse Grids; but ultimately, the science teams must select the best resources for their application. Maintaining and validating the accuracy of a simulation requires programmer resources, and often application teams narrow their efforts to a handful of favorite platforms and sites. Additionally, these sites should have their urgent computing priority policy in place and clearly defined. In the same way that emergency equipment, personnel, and procedures are periodically tested for preparedness and flawless operation, SPRUCE proposes to have applications and policies in warm standby mode, being periodically tested and their date of last validation logged (see Figure 3).

From this pool of warm standby Grid resources, the team must identify where to submit their urgent jobs. In a distributed Grid of independent resource providers, different urgent computing policies will exist. For example, one site may provide only a slightly increased priority to SPRUCE jobs, while another site may kill all the running jobs and allow an extremely urgent computation to use an entire supercomputer. On resources where existing jobs are not killed or preempted, current job load will affect resource selection. For urgent applications that produce or consume massive amounts of data, data movement may also place constraints on resource selection. Moreover, how a given

Platform	App.	Validated	Reliability	...
Site 1::Resource 1	Tornado	8 days ago	95%	...
Site 1::Resource 2	City Airflow	14 days ago	98%	...
Site 2::Resource 1	City Airflow	45 days ago	78%	...
Site 2::Resource 1	Influenza	30 days ago	59%	...

Figure 3. Example of the warm standby log, which tracks the validation history of urgent applications on specific resources

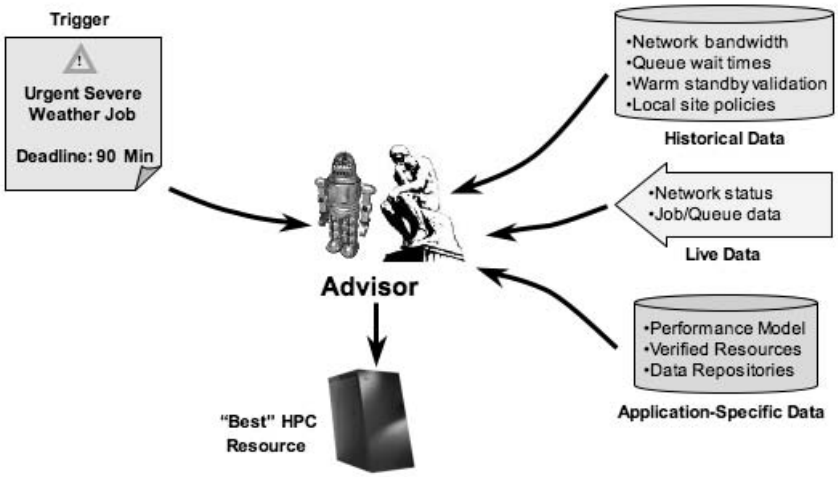


Figure 4. The SPRUCE advisor helps choose the best resource.

application performs on each of the computational resources must also be considered. To support resource selection, SPRUCE users may select resources manually or may use an automated SPRUCE "advisor" (see Figure 4).

The SPRUCE advisor, currently under development, must determine which resources offer the greatest probability to meet the given deadline. To accomplish this task, the advisor must consider a wide variety of information, including the deadline, historical information (e.g., warm standby logs, local site policies), live data (e.g., current network/queue/resource status), and application-specific data (e.g., the set of warm standby resources, performance model, input/output data repositories). To determine the likelihood of an urgent computation meeting a deadline on a given resource, the advisor must calculate an upper bound on the total turnaround time for the job (i.e., the amount of time from when a job is submitted to when the output is ready to be analyzed). Generally, the turnaround time can be divided into three parts: the data movement for input and output data-sets, the resource acquisition delay before the computation begins execution, and the time executing the computation.

The data movement delay includes the time to stage data in as well as the time it takes to transfer output to a destination for analysis. There are several approaches to predicting large file transfer delay by combining sporadic, historic transfer delay data with frequent, lightweight network probe data [5,6]. Additionally, recent work [7] incorporated into the Network Weather Service (NWS) [8] implements two strategies for determining an upper bound on the expected network bandwidth. Both approaches make

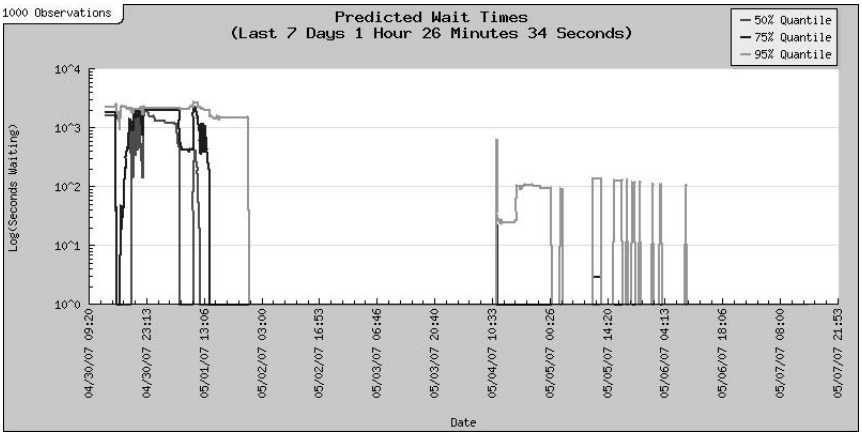


Figure 5. Time series display of the queue delay predictions for next-to-run jobs requesting 17 to 32 nodes on the UC-TeraGrid IA-64 cluster (image courtesy Rich Wolski, UCSB).

forecasts based upon a historical time series, generated via suitably sized NWS network probes that estimate the available bandwidth between the source and destination.

In general, the resource acquisition delay represents the time a job waits prior to execution. For SPRUCE, this delay entirely depends upon the requested urgency and the local policy for each resource. Since each resource provider implements a local policy for handling urgent computation, it becomes a key variable for determining the bound on the queue delay. A prototype implementation of a bounds-predictor for next-to-run jobs has been created by the NWS team. The predictor analyzes the batch queue logs of a particular resource and generates a bound with a given level of confidence on the amount of time a next-to-run job will wait in the queue. An example of these bounds is depicted in Figure 5. Current research is geared toward determining similar methods for bounds calculations for other resource policy choices, such as preemption and checkpoint/restart.

Ideally, the execution time will be predicted from a performance model of the application on a given resource. If no performance model is available, a prediction may be generated by the warm standby validation logs that track the performance of certain configurations of the application on all warm standby resources.

The total turnaround time can then be bounded by combining the three predicted delays. Additionally, the advisor can consider other aspects that affect the turnaround time, for example, "Can the likelihood of meeting a deadline be improved by increasing or reducing the number of CPUs the urgent job requests?" Increasing the number of CPUs used in the computation should decrease the execution time but may result in an increase in the queue delay.

The turnaround time is not the only factor a user must consider when selecting a resource. The reliability of the application on a given resource is also important. For instance, the user may choose a resource that has a slower predicted turnaround time but on which the application has performed more reliably. As part of the warm standby mechanism, SPRUCE can create a reliability metric based on historical validation tests. Also, the current queue state for each resource (which can be provided via MDS4 [9] on the TeraGrid) can be analyzed. The current queue delay methodology predicts bounds based on historical queue data for each resource, and not on the current state of the queue.

However, the user may be able to detect certain situations that may impact their decision, such as an idle resource or a resource that just began executing a long-running job that is consuming the entire resource.

An important question is how to handle job failures. When successful completion of an urgent job is paramount, it may be prudent to simultaneously submit the job to multiple resources. Not only is the likelihood of successful completion increased, but the results may also be compared for validation. In the event that a resource fails while a job is in the queue, the job should be automatically routed to another resource. This resource may be selected by the advisor or specified a priori by the user. If the job fails or the machine goes down during execution, the user should be supplied with a list of alternate resources. We propose this approach rather than automatic rerouting, for two reasons. First, enough time might have elapsed that it is no longer possible to meet the deadline on any available resource; as a result, the computation would be wasted. Second, the execution may have produced partial results that could be used to guide a subsequent computation.

3.4. Prioritized Job Submission

SPRUCE provides support for both Globus-based urgent submissions and direct submission to local job-queueing systems. Currently SPRUCE supports all the major resource managers such as Torque, LoadLeveler, and LSF and schedulers such as Moab, Maui, PBS Pro, and Catalina. The system can be extended to any scheduler with little effort. Authorized users who have active tokens need only to specify an additional "urgency" parameter when submitting their jobs.

The Globus Toolkit [10] for Grid computing provides the TeraGrid with uniform tools for authentication, job submission, file transfer, and resource description. By extending the Resource Specification Language (RSL) [11], which is used by Globus to identify user-specific resource requests, the ability to indicate a level of urgency for jobs is incorporated. A new "urgency" parameter is defined for three levels: *critical* (red), *high* (orange), and *important* (yellow). These urgency levels are guidelines that help resource providers enable varying site-local response protocols to differentiate potentially competing jobs. Users with valid SPRUCE tokens can simply submit their original Globus submission script with one additional RSL parameter (of the form 'urgency = <level>'), to gain priority access.

Unlike the Globus RSL, local job queue submission interfaces, such as the PBS command *qsub* [12], are often not trivially extended to accept new parameters. Specification of urgency level when submitting directly to a resource's local job queue typically requires a modified job submission command or a wrapper script. SPRUCE provides a *spruce_sub* script that accepts an additional command line parameter specifying the job's requested urgency level.

From the user's perspective, the next step after submitting the job is to analyze the results once the job successfully completes execution. On the computing resource, behind the scenes, the job script is parsed, and a check is performed with the SPRUCE server to verify the user has permission to run urgent jobs before taking the corresponding action (see Section 3.5.3).

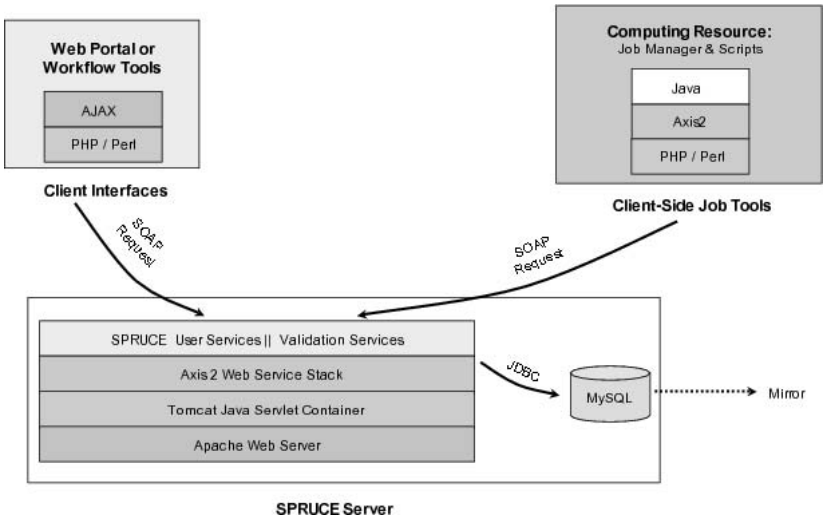


Figure 6. Internal configuration of the SPRUCE portal

3.5. SPRUCE Portal

The SPRUCE portal provides a single-point of administration and authorization for urgent computing across an entire Grid. It consists of three parts:

- The Web-based administrative interface allows privileged site administrators to create, issue, monitor, and deactivate right-of-way tokens. It features a hierarchical structure, allowing management of specific sub-domains.
- The Web service-based user interface permits token holders to activate an urgent computing session and to manage user permissions.
- The authentication service verifies urgent computing job submissions. A local site job manager agent queries the remote SPRUCE server to ensure that the submitting user is associated with an active token that gives permission to run urgent jobs on the given resource at the requested urgency.

The internal architecture of the SPRUCE portal is depicted in Figure 6. Both the user interface and the authentication service communicate with the SPRUCE server via a Web services interface. The user interface is implemented on top of Apache Axis 2 [20], while the portal is implemented in PHP and uses MySQL. External portals and workflows can become SPRUCE-enabled simply by incorporating the necessary Web service invocations.

SPRUCE users can interact with the system using a Web browser, requiring only minimal additional training, and making SPRUCE appropriate for emergency situations. Likewise, administrators will find the interface easy to navigate and use regardless of their environment.

3.5.1. Administration Interface

Distributed Grids typically span multiple administration domains. For the TeraGrid, the Grid Infrastructure Group (GIG) coordinates the software infrastructure, allocation, us-

age reporting, user support, and Grid security. While each resource provider, such as the San Diego Supercomputer Center or the University of Chicago, has a defined "service level agreement" for participation in the TeraGrid, they are nevertheless independent organizations. To support multiple administrative domains and virtual organizations, SPRUCE maintains a hierarchical Web-based administration interface, organized into three domains. Ordered by increasing privileges, these domains are the site (Grid resource provider), virtual organization, and root administrator.

All administrators have the ability to create and distribute tokens that are limited to particular resources and levels of priority within their domain, along with the privilege to revoke any tokens they created. They also have access to the token activation and urgent job submission history, user logs, and related statistics for their respective domain.

To permit distinct resource and management policies at each site, SPRUCE maintains sites within a virtual organization as independent entities. This strategy enables the site administrator to use SPRUCE in the wider context of a large multisite Grid, as well as privately for local machines and users. A virtual organization, such as the TeraGrid, spans multiple sites. An administrator in this domain may issue tokens that are valid across any sites that are a part of the virtual organization. Additionally, virtual organization administrators can add new sites. The most privileged administrative domain is the SPRUCE super-user, who is responsible for managing the virtual organizations. We note that SPRUCE does not support tokens that span multiple virtual organizations. The reason is that one of the defining features of a virtual organization is that their users are uniquely identified (e.g., certificate-based Distinguished Name, UNIX user name). However, sites belonging to distinct virtual organizations that utilize different user identity mechanisms and wish to allow cross-site tokens may form a new virtual organization with a shared identity mechansim.

3.5.2. User Interface

The intended users of the SPRUCE Web interface are scientists responsible for organizing the application team. Their tasks include monitoring the status of tokens, activating sessions, and organizing the team that will participate in an urgent computing session. This interface must be simple, fast, and modeled after the workflow described earlier in Section 3.2.

The Web services architecture enables SPRUCE integration with existing scientific Web portals and workflows. Users who prefer to use a Web-based interface can use the SPRUCE user portal. When a token is activated, the urgent computing session begins immediately; it terminates once the lifetime of the token has lapsed. We note that the session lifetime refers only to the period when urgent jobs can be submitted. Once the session ends, jobs that are currently executing will be allowed to continue uninterrupted. For convenience, members of the team who can submit jobs can be organized (added or removed) before the token is activated or any time during an active session. Changes to the set of users associated with a token are propagated without delay. All SPRUCE users may monitor basic statistics such as the remaining lifetime of the token and can query SPRUCE to find out the tokens with which they are currently associated.

3.5.3. Job Authentication Interface

At the core of the SPRUCE architecture is the invariant that urgent jobs may be submitted only while a right-of-way token is active. In order to support this invariant across a

distributed Grid system, a remote authentication step must be inserted into the job submission tool-chain for each resource supporting urgent computation. Since the SPRUCE portal contains the updated information regarding active sessions and users permitted to submit urgent jobs, it is also the natural point for authentication.

When an urgent computing job is submitted via Globus or the local queue system, the urgent priority parameters trigger authentication. This authentication is not related to a user's access to resource, which has already been handled by the traditional Grid certificate or by logging into the Unix-based resource. Rather, it is a "Mother, may I" request for permission to enqueue a high-priority job. This request is sent to the SPRUCE portal, where it is checked against active tokens, resource names, maximum priority, and associated users. Permission is granted if an appropriate right-of-way token is active and the job parameters are within the constraints set for the token. All transactions, successful and unsuccessful, are logged.

3.6. Resource Providers

To provide urgent computing capabilities for a supercomputer with SPRUCE, the resource providers must take three actions: register with the SPRUCE portal, formulate a resource specific policy for responding to urgent computing requests, and install SPRUCE components that interface with the job manager and the queueing system.

3.6.1. Portal Registration

Sites that use SPRUCE need an administrative account on the SPRUCE portal. Using this account, administrators can specify the details for each of the computational resources supporting urgent job submissions. The site administrator will also provide important contact information that can be used for emergency notifications when tokens are activated or critical errors occur. Once a site is registered, the administrator may begin generating and issuing right-of-way tokens. If the site is a member of a larger distributed Grid system that is already a part of SPRUCE, it may be incorporated into the corresponding virtual organization.

3.6.2. Responding to Urgent Computation

The SPRUCE architecture does not define or assume any particular policy for how sites respond to urgent computing requests. This approach complicates some usage scenarios, but it is unavoidable given the way we build Grids from distributed resources of independent autonomous centers, and the diversity of resources and operating systems available for computing.

When small-memory vector computers were the standard for HPC computing, preempting jobs was natively supported. Long-running jobs were routinely suspended, not to support urgent decision calculations, but to permit shorter jobs to achieve fast turnaround times during compile or debug sessions. Unfortunately, almost all modern supercomputers have lost this key feature, and therefore the SPRUCE architecture cannot simply standardize the strategy for responding to urgent computation as immediate preemption. Instead, we are left with many possible choices for supporting urgent computation depending on the systems software and middleware as well as on constraints based on accounting of CPU cycles, machine usability, and user acceptance. Given the current

technology for Linux clusters and more tightly integrated systems such as the Cray XT3 and the IBM Blue Gene, the following responses to an urgent computing request are possible:

- Scheduling the urgent job as "next-to-run" in a priority queue. This approach is simple and is highly recommended as a possible response for all resource providers. All modern queueing and job management systems support priority queues that are used for selecting the next job to run. No running computation is killed; and from the perspective of the user community, the impact on normal use is low. The urgent job will begin when all of the running jobs complete for a given set of CPUs.
- Suspending running jobs and immediately launching the urgent job. Some systems allow jobs to be suspended but remain resident in memory (via STOP signal). Running the urgent job will then force some memory paging, but the suspended job could be resumed later. Some applications that use external data sources and network connections may fail (connections time out and reset) if they are suspended. If a node crashes, both the suspended and the urgent jobs will be lost. The benefit of this policy is that urgent computation will begin almost immediately, making this option very attractive in some cases.
- Forcing a checkpoint/restart of running jobs and enqueueing the urgent job as the next to run. This response is similar to the previous response but safely moves the checkpoint to a location where it can then be used to restart on alternate resources, and is safe from node reboots. Some architectures support system-based checkpoint/restart; and, where it is reliable, it could be used to support urgent jobs. The urgent job begins execution when the checkpoint completes, which for large-memory systems could take 30 minutes or more depending on I/O and disk rates.
- Killing all running jobs and enqueueing the urgent job as next to run. Clearly this response is drastic and frustrating to the users who will lose their computation. Nevertheless, for extremely urgent computation, what user would demand that a black-hole simulation complete before launching an emergency hurricane or flood modeling scenario? In this response, an urgent job would begin immediately after running jobs are killed. Often, the delay to start is only several minutes.

Another factor in choosing the response policy is accounting and stakeholder accountability. Certain machines are funded for specific activities, and only a small amount of discretionary time is permitted. In some cases, there may be no specific "charge code" for urgent computing cycles. Furthermore, in order to improve fairness, some form of compensation could be provided to jobs that are killed to make room for an urgent job. For example, users could be refunded their CPU hours, given extra time for their trouble, and rescheduled with higher priority in order to avoid being relegated to the back of the job queue.

Another idea is to provide discounted CPU cycles for jobs that are willing to be terminated to make room for urgent computations. Some users have extremely robust and well-integrated problem solving environments that can perform checkpoint/restart easily. Some users design their software so only one or two hours of work is lost should a CPU fail or the entire system go down. Such users should be rewarded, and a discounted rate would allow them to recover lost work and run more inexpensively.

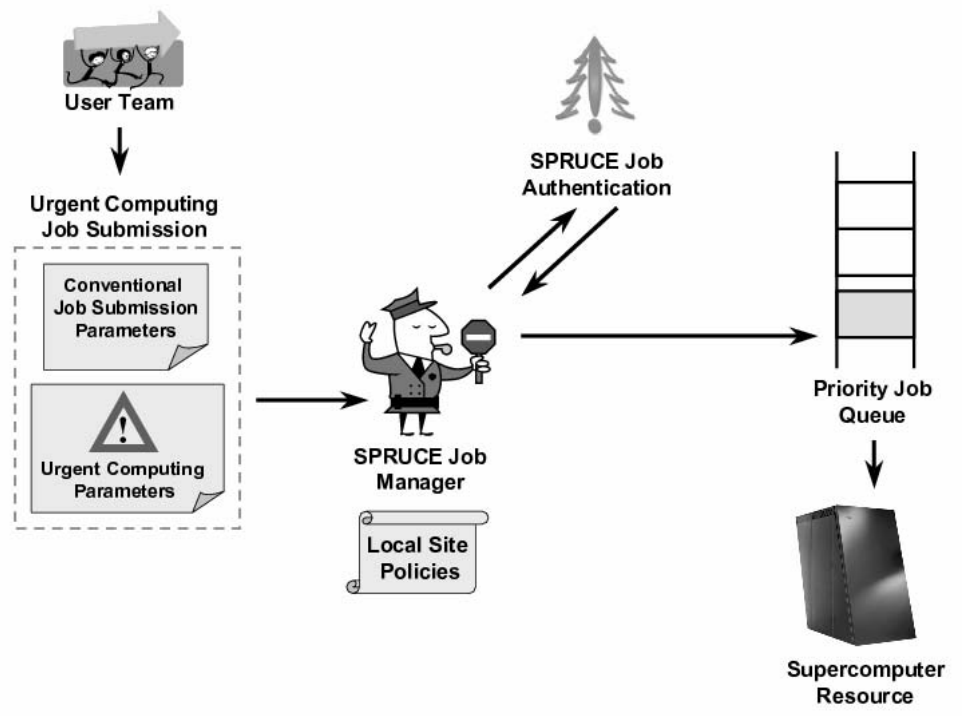


Figure 7. Resource provider architecture

The calculation of "maximum time to begin" (i.e., resource acquisition delay) discussed in Section 3.3 may play an important role in choosing a response strategy. For checkpoint/restart and killing of running jobs, the maximum time to begin can be bounded, possibly on the order of a few minutes to tens of minutes. If it is easy to calculate or determine, it can be used in conjunction with the computation deadline for selecting resources. Unfortunately, jobs with next-to-run priority could wait hours or days before running jobs complete. In any case, resource providers are encouraged to map all three levels of urgency—critical, high, and important—to clearly defined responses.

3.6.3. Handling Urgent Job Submissions

Figure 7 gives an overview of how the job requests are handled at the resource provider. In the Globus architecture, incoming jobs are routed through a job manager. When SPRUCE is installed on the resource, a job manager tailored to handle the additional urgency job parameters is added. This generates a job script dynamically when the urgent job is submitted and passes it to the native resource manager such as PBS Pro or Torque. It then authenticates the request against the SPRUCE portal (see Section 3.5.3). For example, in the case of the Torque scheduler, a submit-filter [13] script specific to SPRUCE is run every time a job is submitted. This filter authenticates the job script and confirms that it was prepared by the job manager rather than a malicious user attempting to submit a job into the high-priority queue without SPRUCE validation. If the user does not have sufficient permission, the request is rejected. If the request passes the verification stage, the actions needed to grant urgent access are performed based on the local site policy

and the requested priority level. After verification, the native job scheduler sends the job ID back to Globus, and when the requested resources become available, the queued job is launched.

Local sites can also support the command line version of the urgency job submission mechanism in the form of *spruce_sub*. Submission requests of this type are also routed through the SPRUCE job manager; the implementation mechanism remains the same. The only difference between these two submission methods is the interface.

3.7. Future Work

The SPRUCE system is currently running in production on many sites. Our current research focus is the automated "advisor" (see Section 3.3). This capability will need some job-specific information such as running time, data dependencies, and other possible computer-specific characterizations that can be collected periodically via verification drills. Work is under way to create a framework that can automatically handle these warm standby runs to verify application and policy readiness. We plan to leverage the INCA monitoring system [14] and MDS4 of the Globus Toolkit for this purpose. An important challenge is to be able to "encode" local site policies in such a way that they can be probed by the advisor.

Another goal is to incorporate more flexibility into job submissions and tokens. Currently, users can specify only the urgency level in their RSL. The inclusion of more information such as deadline can help the advisor select a suitable resource for the job to run. Some applications, such as hurricane and tornado modeling, have a lead time before they will need priority access. In such cases, the policy would need to be able to clear off the queues within the lead time, rather than providing next-to-run or preemptive access. Additionally, token holders might want the ability to aggregate tokens, rather than manage back-to-back separate sessions.

Yet another aspect that we are considering is providing a priority resubmit coupon to users whose jobs have been preempted during an urgent session. This coupon is similar to a token but would be valid for a single submission that is constrained to be similar to the preempted job (e.g., similar number of requested CPUs, requested wallclock time, etc.). The coupon would allow users to return to the top of the queue rather than the bottom, where they may wait days before getting back on the machine. This strategy provides a straightforward bound on the delay incurred when users are preempted - they will be inconvenienced no longer than is required for the urgent job to complete.

Currently, SPRUCE does not provide tools for prioritized data movement, which is crucial for many high-performance computations. Existing data movement strategies and tools such as the GridFTP project [15] will facilitate SPRUCE in the future. In addition, network provisioning that reserves bandwidth may be needed for applications with large data requirements [16].

One drawback of the current design is that there exists a single point of failure - the SPRUCE portal. If the portal is experiencing downtime or the user fails to access it, the only alternative is to abort an urgent submission. In order to counteract this weakness, the portal and the backend database will require redundancy and remote fail-over locations. The existing version of the portal is also subject to the same variety of attacks as other Internet Web servers, including denial of service, spoofing, and abuse of software vulnerabilities. These and other research topics have received considerable attention; we hope to take advantage of these efforts in our future work.

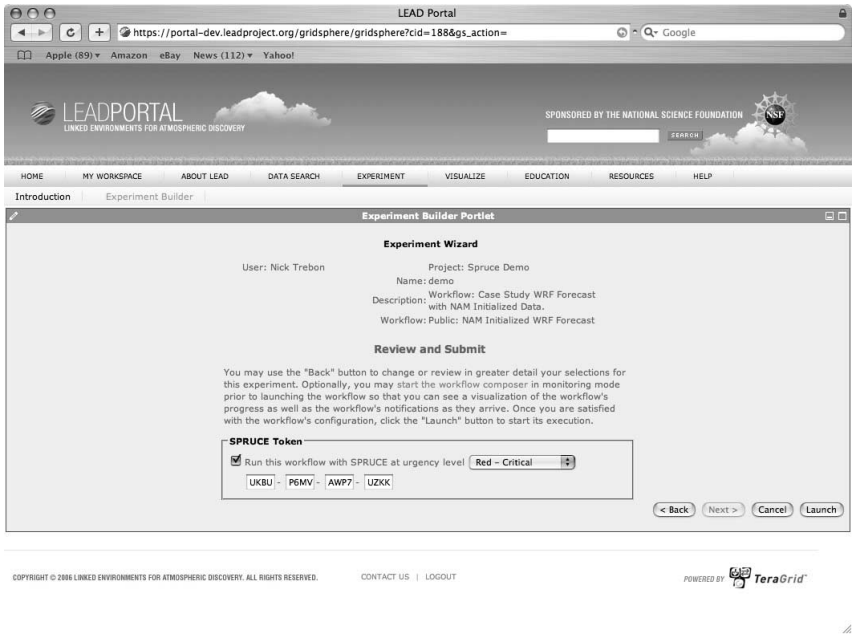


Figure 8. Screenshot of SPRUCE integrated into the LEAD automated workflow

3.8. Experiences

Currently, SPRUCE is deployed at the University of Chicago/Argonne National Laboratory (UC/ANL), Purdue University, Texas Advanced Computing Center (TACC), San Diego Supercomputer Center (SDSC), and the National Center for Supercomputing Applications (NCSA) and is being deployed at Indiana University. Virginia Tech University is one of our early non-TeraGrid adopters planning to use this system for simulating epidemiological spread patterns as a part of EPISIMS [17]. Louisiana State University is also considering installing SPRUCE on it's local cluster for the SCOOP [18] project.

Collaboration between the LEAD weather forecasting and analysis project [19] and the UC/ANL TeraGrid resources resulted in real-time, on-demand severe weather modeling and forecasting during May and June 2007. Additionally, the UC/ANL IA64 machine currently supports preemption for the highest priority urgent jobs. As an incentive to use the platform even though jobs may be killed, users are charged at a rate of 90 percent of the standard CPU service unit billing. Deciding which jobs are preempted is determined by an internal scheduler algorithm that considers several aspects, such as the elapsed time for the existing job, number of nodes and jobs per user. The complete policy mapping on the IA64 UC/ANL resource is as follows.

- Yellow - elevated priority
- Orange - next-to-run status
- Red - preemptive access

The LEAD collaboration clearly demonstrates the capability that SPRUCE can provide to existing workflows. LEAD was given a limited number of tokens for use throughout the tornado season. Urgent runs are triggered automatically by parsing the RSS feed

of advisories published by National Weather Service for possible warning flags. A right-of-way token at the required urgency level is activated based on the warning, and a list of users is added onto the token automatically via the Web service interface to the SPRUCE server. Alternatively, community users can activate tokens given to them from the LEAD portal directly, without logging into the SPRUCE portal (see Figure 8).

A significant challenge involved allowing local sites to establish their own policies while keeping the SPRUCE installation as simple as possible. Each site needs a customized version of the job manager (dependent on the site policy and scheduler), which cannot be bundled into a common distribution. Hence, site administrators must make minor modifications to the distributed SPRUCE job manager to integrate SPRUCE into their systems. All these changes are well documented.

4. Discussion

Apart from the framework, urgent computing needs to focus on three key areas in the long run: Policy, Planning and Technology.

4.1. Policy

Most resource providers and users support the basic ideas of on-demand and urgent computing. However, the realities of resource utilization, user expectations, and current policies make supporting urgent computing challenging. At this time, funding agencies such as the NSF and DOE evaluate HPC centers on utilization and delivered flops per dollar. Viewed in this narrow scope, support for urgent computing has few benefits to the participants in this market. Warm standby computing drills and eager scheduling of urgent jobs depress resource utilization. Urgent runs can disrupt running jobs and delay those waiting in the queue. Drill runs to provide application warm standby are also costly, with no clear answer as to which allocation should be charged for the continual testing of an urgent application. However, these issues stem from the priorities and metrics used by the stakeholders. Traditionally, the General Accounting Office has asked large government-funded supercomputer centers for measures including system utilization, cost, and average time waiting in a queue. Hand-in-hand with building support for urgent computing we must create new metrics and policies to reflect priorities other than peak flops per dollar.

Because urgent computing is by its very design disruptive to the normal operation of a center, incentives and pricing models must be explored to encourage user participation and adoption. For example, urgent computing can create incentives for application developers to develop codes that can be checkpoint/restarted or preempted. Providing rewards or CPU-pricing incentives to use special “interruptible” job queues can help free more resources for urgent computing. In return, jobs from these queues will be offered a discounted rate. There is also an implicit benefit for applications to be compatible with urgent computing because it encourages development of code that is robust to interruptions and failures.

Another challenge is the human response to disruptions caused by urgent computing. For example, SPRUCE changes the established model of resource control. In handing out a SPRUCE token to a resource, the administrator is making a promise to a po-

tential future computation. The administrator is also delegating the decision to preempt jobs to someone else. It removes some of the hands-on control administrators are used to. Outstanding SPRUCE tokens increase the probability of an urgent computing job disrupting regular use of the resource. Since urgent computations are unplanned, they also jeopardize resource utilization in the long run. To participate in an urgent computing network, the rewards for the resource provider must outweigh the reduced utilization and decreased predictability of the system.

4.2. Planning

During security breaches, professionals rely on playbooks to dictate a swift course of action. Likewise, emergencies require fast and thoughtful response based on early planning and organization. Playbooks, or step-by-step instructions on a course of action during emergency scenarios, are especially valuable in this regard. Urgent computing is not something that can be organized on the fly, and we believe that resource providers, urgent application programmers, scientists, and decision makers should all maintain urgent computing playbooks for preparedness. Besides thorough instructions, an urgent computing playbook can contain essential contact information for all the participants in a scenario response to mitigating the overhead of organizing a team during an emergency. Playbooks can also increase awareness of how applications and their output contribute to a coherent response plan that may span multiple organizations, and many time zones.

For many situations where urgent computing could provide insight, the set of relevant codes can be organized in advance. However, there will always be situations outside the planned scenarios or playbooks. Locating application codes and science teams relevant to a particular scenario can take a prohibitively long time. During emergencies, officials should know what relevant applications and scientists might be able to help guide a response scenario. To facilitate this process, we envision a database that aggregates information about applications that could be useful for constructing a dynamic response.

4.3. Technology

Virtualization technologies such as VMWare [21] and Xen [22] hold considerable promise for urgent computing. Virtualization has the benefit of standardizing a runtime environment that can be transported to a resource without worrying about an application's library and architecture dependencies. Urgent codes maintained in virtual machines can be deployed to the resource and begin execution immediately without any overhead of manual configuration. This will lower turnaround time and simplify application maintenance and drill procedures. Another benefit of using virtual machines is that they minimally impact already running virtualized applications through graceful degradation. In a virtualized environment, checkpointing or killing a running job is not necessary to simultaneously execute an urgent job, since the urgent job can time share the CPU and other resources. Or, the virtual machine for a preempted job could be moved to a smaller or slower resource. One drawback to using virtual machines, however, is their large memory and disk footprint. A virtual machine image needs to include libraries, configuration settings, and other data that is usually not bundled with the application.

In building and deploying SPRUCE we have concentrated our efforts primarily on federally sponsored supercomputing centers. However, there are commercial alternatives

to computing and storage resources. For example, Amazon provides the EC2 (Elastic Compute Cloud) service [23] and S3 (Simple Storage Service) [24] frameworks. Although they are "best effort" resources, these commercial services can be potential resources during urgent need, especially in cases when other infrastructure is unavailable. Amazon's services are competitive alternatives because, unlike other resource providers, they fully support virtualization and can prioritize and coordinate large reservations for customers who are willing to pay the premium.

5. Conclusions

Urgent computing is a new and evolving field made possible by the improved fidelity and utility of high-performance computing to decision making and near-real-time instrument steering and control. To support this new field, scientists must work together to develop not only the technology but also the policies to support research and development for emerging urgent computing applications. The community must organize policies that can be used to help guide answers to the questions of "who," "why," and "under what circumstances" application teams will receive permission to initiate urgent computing sessions. To build an international infrastructure to support urgent computing, we must solve both challenges: political and technical. We believe that by deploying prototype systems such as SPRUCE, the community can test and explore this new way to use applications and resources.

6. Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

References

[1] "IPCC Summary for Policymakers," http://www.ipcc.ch/SPM040507.pdf.
[2] National Aeronautics and Space Administration, "High-End Computing at NASA," Technical Report, 2006.
[3] "Urgent Computing Workshop 2007," http://spruce.uchicago.edu/workshop/urgent07.php.
[4] "Telecommunications Service Priority (TSP) program," http://tsp.ncs.gov.
[5] S. Vazhkudai and J. Schopf, "Using Regression Techniques to Predict Large Data Transfers," in *International Journal of High Performance Computing Applications*, 2003, pp. 249–268.
[6] M. Swany and R. Wolski, "Multivariate Resource Performance Forecasting in the Network Weather Service," in *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002, pp. 1–10.
[7] R. Wolski, "Experiences with Predicting Resource Performance On-Line in Computational Grid Settings," *ACM SIGMETRICS Performance Evaluation Review: SPECIAL ISSUE: Special section on Grid computing*, 2003, pp. 41–49.
[8] R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *Future Generation Computer Systems*, pp. 757–768, 1999.
[9] J. Schopf, M. D'Arcy, N. Miller, L. Pearlman, I. Foster, and C. Kesselman, "Monitoring and Discovery in a Web Services Framework: Functionality and Performance of the Globus Toolkit's MDS4," Argonne National Laboratory, Preprint ANL/MCS-P1248-0405, April 2005.

[10] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," in *IFIP International Conference on Network and Parallel Computing*, 2005, pp. 2–13.
[11] "Globus Resource Specification Language," http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html.
[12] "PBS 'qsub' Job Submission Tool," http://www.clusterresources.com/torquedocs21/commands/qsub.shtml.
[13] "Torque Submit Filter," http://www.clusterresources.com/products/torque/docs20/a.jqsubwrapper.shtml.
[14] "Test Harness and Reporting Framework (INCA)," http://inca.sdsc.edu.
[15] "GridFTP Project," http://www.globus.org/grid_software/data/gridftp.php.
[16] L. Gommans, F. Travostino, John, Vollbrecht, C. de Laat, and R. Meijer, "Token-based Authorization of Connection Oriented Network Resources," in *GRIDNETS Conference Proceedings*, October 2004.
[17] "EPISIMS and Simfrastructure," http://ndssl.vbi.vt.edu/episims.html.
[18] "Sura Coastal Ocean Observing and Prediction," http://www.scoop.lsu.edu/gridsphere/gridsphere.
[19] "Linked Environments for Atmospheric Discovery (LEAD)," http://lead.ou.edu/.
[20] "Apache Axis 2," http://ws.apache.org/axis2/.
[21] "VMWare,", http://www.vmware.com/.
[22] "Xen Virtualization Software," http://www.xensource.com/.
[23] "Amazon EC2, Amazon Elastic Compute Cloud," http://aws.amazon.com/ec2.
[24] "Amazon S3, Amazon Simple Storage Service," http://aws.amazon.com/s3.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Network Communication as a Service-Oriented Capability

William JOHNSTON, Joe METZGER, Mike O'CONNOR, Michael COLLINS, Joseph BURRESCIA, Eli DART, Jim GAGLIARDI, Chin GUOK and Kevin OBERMAN
ESnet, Lawrence Berkeley National Laboratory

Abstract. In widely distributed systems generally, and in science-oriented Grids in particular, software, CPU time, storage, etc., are treated as "services" – they can be allocated and used with service guarantees that allows them to be integrated into systems that perform complex tasks. Network communication is currently not a service – it is provided, in general, as a "best effort" capability with no guarantees and only statistical predictability.

In order for Grids (and most types of systems with widely distributed components) to be successful in performing the sustained, complex tasks of large-scale science – e.g., the multi-disciplinary simulation of next generation climate modeling and management and analysis of the petabytes of data that will come from the next generation of scientific instrument (which is very soon for the LHC at CERN) – networks must provide communication capability that is service-oriented: That is it must be configurable, schedulable, predictable, and reliable. In order to accomplish this, the research and education network community is undertaking a strategy that involves changes in network architecture to support multiple classes of service; development and deployment of service-oriented communication services, and; monitoring and reporting in a form that is directly useful to the application-oriented system so that it may adapt to communications failures.

In this paper we describe ESnet's approach to each of these – an approach that is part of an international community effort to have intra-distributed system communication be based on a service-oriented capability.

Keywords. Energy Sciences Network (ESnet), networks for large-scale science, network planning, networks and service oriented environments

1. The Network Today

1.1. ESnet's Mission

ESnet's mission is to provide an interoperable, effective, reliable, high performance network communications infrastructure, along with selected leading-edge Grid-related and collaboration services in support of the large-scale, collaborative science that is integral to the mission of DOE's Office of Science (SC).

ESnet must provide services that enable the SC science programs that depend on:

- Sharing of massive amounts of data
- Supporting thousands of collaborators world-wide
- Distributed data processing
- Distributed data management

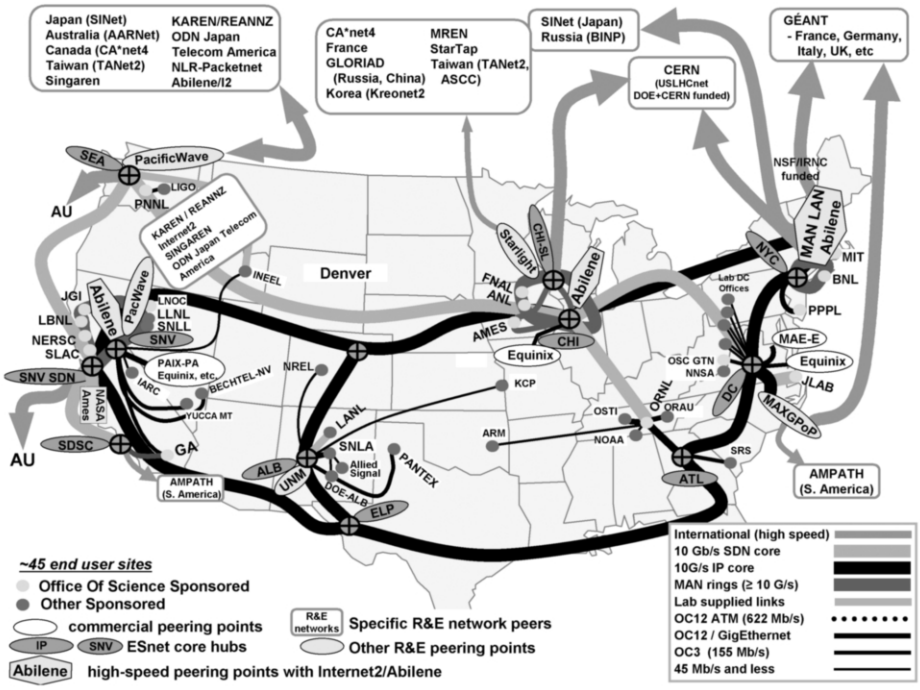


Figure 1. ESnet provides global high-speed Internet connectivity for DOE facilities and collaborators (ESnet in early 2007).

- Distributed simulation, visualization, and computational steering
- Collaboration with the US and International Research and Education community.

To this end, ESnet provides network and collaboration services to DOE laboratories. ESnet also serves programs in most other parts of DOE.

1.2. ESnet Defined

ESnet is:

- A large-scale IP network built on a national circuit infrastructure with high-speed connections to all major US and international research and education (R&E) networks.
- An organization of 30 professionals structured for the service.
- An operating entity with an FY06 budget of $26.6M.
- A tier 1 ISP providing direct peering will all major networks – commercial, government, and research and education (R&E).
- The primary DOE network providing production Internet service to almost all of the DOE Labs and most other DOE sites. This results in ESnet providing an estimated 50,000–100,000 DOE users and more than 18,000 non-DOE researchers from universities, other government agencies, and private industry that use SC facilities with global Internet access.

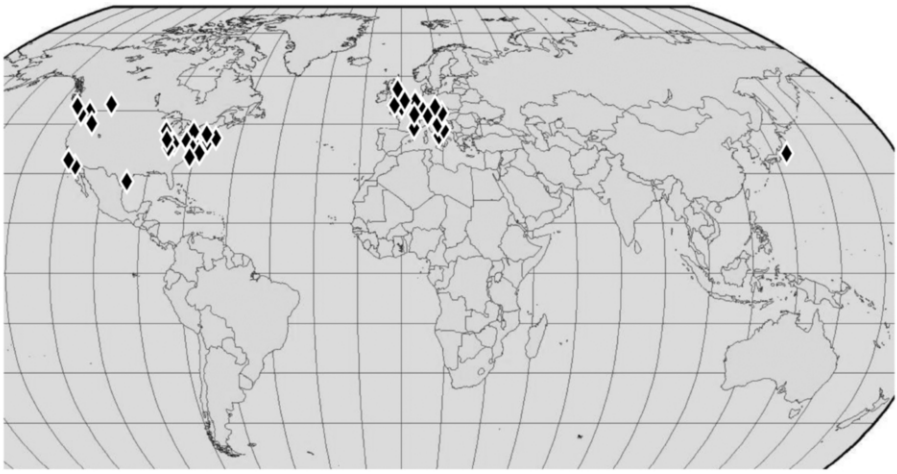


Figure 2. The large-scale data flows in ESnet reflect the scope of Office of Science collaborations. ESnet's top 100 data flows generate 50% of all ESnet traffic (ESnet handles about 3×10^9 flows/mo.) 91 of the top 100 flows are from the DOE Labs (not shown) to other R&E institutions (shown on the map) (CY2005 data).

1.3. ESnet's Place in U.S. and International Science

A large fraction of all of the national data traffic supporting U.S. science is carried by three networks – ESnet and Internet2, and National Lambda Rail. These three entities fairly well represent the architectural scope of science-oriented networks.

ESnet is a network in the traditional sense of the word. It connects end-user sites to various other networks. Internet2 is primarily a backbone network. It connects U.S. regional networks to each other and International networks. NLR is a collection of light paths or lambda channels that are used to construct specialized R&E networks.

ESnet serves a community of directly connected campuses – the Office of Science Labs. In essence ESnet interconnects the LANs of all of the Labs to the outside world. ESnet also provides the peering and routing needed for the Labs to have access to the global Internet. Internet2 serves a community of regional networks that connect university campuses. These regional networks – NYSERNet (U.S. northeast), SURAnet (U.S. southeast), CENIC (California), etc., – have regional aggregation points called GigaPoPs and Internet2 interconnects the GigaPoPs. Internet2 is mostly a transit network – the universities and/or the regional networks provide the peering and routing for end-user Internet access. This is very similar to the situation in Europe where GÉANT (like Internet2) interconnects the European National Research and Education Networks (NRENs) that in turn connect to the LANs of the European science and education institutions. (The NRENs are like the US regional networks, but organized around the European nation-states).

The top-level networks – ESnet, Internet2, GÉANT, etc. – work closely together to ensure that they have adequate connectivity with each other so that all of the connected institutions have high-speed end-to-end connectivity to support their science and education missions. ESnet and Internet2 have had joint engineering meetings for several years (Joint Techs) and ESnet, Internet2, GÉANT, and CANARIE (Canada) have also formed an international engineering team that meets several times a year.

An ESnet goal is that connectivity from DOE Lab to US and European R&E institutions should be as good as Lab to Lab and University to University connectivity. The

key to ensuring this is engineering, operations, and constant monitoring. ESnet has worked with the Internet2 and the international R&E community to establish a suite of monitors that can be used to continuously check a full mesh of paths through all of the major interconnection points.

2. Next Generation Networks

2.1. Evolving Science Environments Drive the Design of the Next Generation ESnet

Large-scale collaborative science – big facilities, massive amount of data, thousands of collaborators – is a key element of DOE's Office of Science. The science community that participates in DOE's large collaborations and facilities is almost equally split between SC labs and universities, and has a significant international component. Very large international (non-US) facilities (e.g., the LHC particle accelerator at CERN in Switzerland and the ITER experimental fusion reactor being built in France) and international collaborators participating in US based experiments are now also a key element of SC science, requiring the movement of massive amounts of data between the SC labs and these international facilities and collaborators. Distributed computing and storage systems for data analysis, simulations, instrument operation, etc., are becoming common, and for data analysis in particular, Grid-style distributed systems predominate. (See, e.g., the Open Science Grid – an SC led distributed Grid computing project – http://www.opensciencegrid.org/.)

This Grid-based science environment is very different from that of a few years ago and places substantial new demands on the network. High-speed, highly reliable connectivity between labs and US and international R&E institutions is required to support the inherently collaborative, global nature of large-scale science. Increased capacity is needed to accommodate a large and steadily increasing amount of data that must traverse the network to get from instruments to scientists and to analysis, simulation, and storage facilities. High network reliability is required for interconnecting components of distributed large-scale science computing and data systems and to support various modes of remote instrument operation. New network services are needed to provide bandwidth guarantees for data transfer deadlines, remote data analysis, real-time interaction with instruments, coupled computational simulations, etc.

There are many stakeholders for ESnet. Foremost are the science program offices of the Office of Science: Advanced Scientific Computing Research, Basic Energy Sciences, Biological and Environmental Research, Fusion Energy Sciences, High Energy Physics, and Nuclear Physics – see http://www.science.doe.gov/. ESnet also serves labs and facilities of other DOE offices (e.g., Energy Efficiency and Renewable Energy, Environmental Management, National Nuclear Security Administration, and Nuclear Energy, Science and Technology). Other ESnet stakeholders include SC-supported scientists and collaborators at non-DOE R&E institutions (more than 85% of all ESnet traffic comes from, or goes out to non-DOE R&E organizations), and the networking organizations that provide networking for these non-DOE institutions.

Requirements of the ESnet stakeholders are primarily determined by three approaches: 1) Instruments and facilities that will be coming on-line over the next 5–10 years and will connect to ESnet (or deliver data to ESnet sites in the case of LHC and IETR) are characterized by considering the nature of the data that will be generated and how and where it will be stored, analyzed, and used. 2) The process of science in

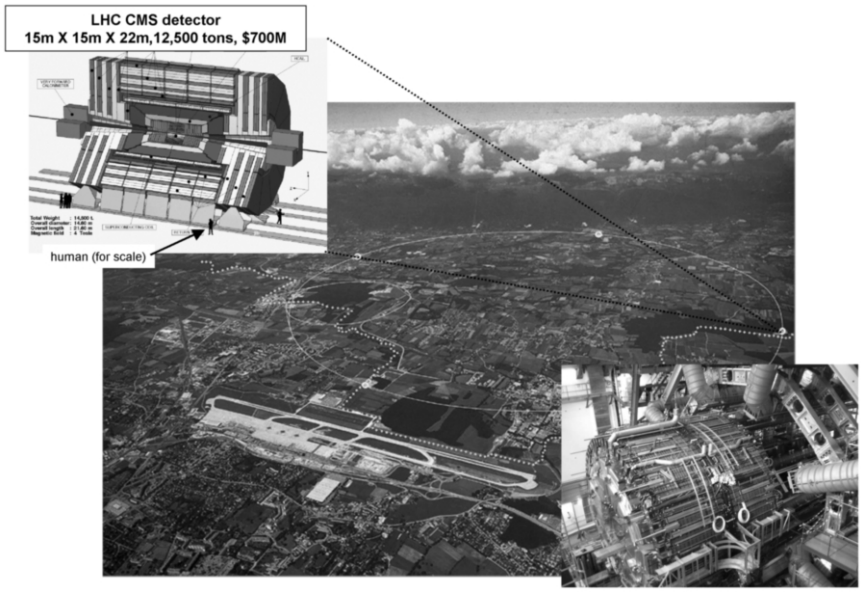


Figure 3. The Large Hadron Collider at CERN. An aerial view of CERN and a graphic showing one of the two large experiments (the CMS detector). The LHC ring is 27 km circumference (8.6 km diameter) and provides two counter-rotating, 7 TeV proton beams collide in the middle of the detectors. (Images courtesy CERN.)

the disciplines of direct interest to SC is examined to determine how the process of that science will change over the next 5–10 years and how these changes will drive demand for new network capacity, connectivity, and services. 3) ESnet traffic patterns are analyzed based on the use of the network in the past 2–5 years to determine the trends, and then projecting this usage forward in time, thus determining how the network must change to accommodate the future traffic patterns implied by these trends.

2.2. A Case Study: The Data Analysis for the Large Hadron Collider [1]

The major high energy physics (HEP) experiments of the next twenty years will break new ground in our understanding of the fundamental interactions, structures and symmetries that govern the nature of matter and space-time. Among the principal goals are to find the mechanism responsible for mass in the universe, and the "Higgs" particles associated with mass generation, as well as the fundamental mechanism that led to the predominance of matter over antimatter in the observable cosmos.

The largest collaborations today, such as CMS [12] and ATLAS [13], which are building experiments for CERN's Large Hadron Collider program (LHC [14]), each encompass some 2000 physicists from 150 institutions in more than 30 countries. The current generation of operational experiments at Stanford Linear Accelerator Center (SLAC) (BaBar [15]) and Fermilab (D0 [16] and CDF [15]), as well as the experiments at the Relativistic Heavy Ion Collider (RHIC, [18]) program at Brookhaven National

[1] Material for this sections is drawn from the "Report of the Standing Committee on Inter-Regional Connectivity (SCIC), Networking for High Energy Physics," February 8, 2007 [7], and from conversations between WEJ and Harvey Newman of Caltech.

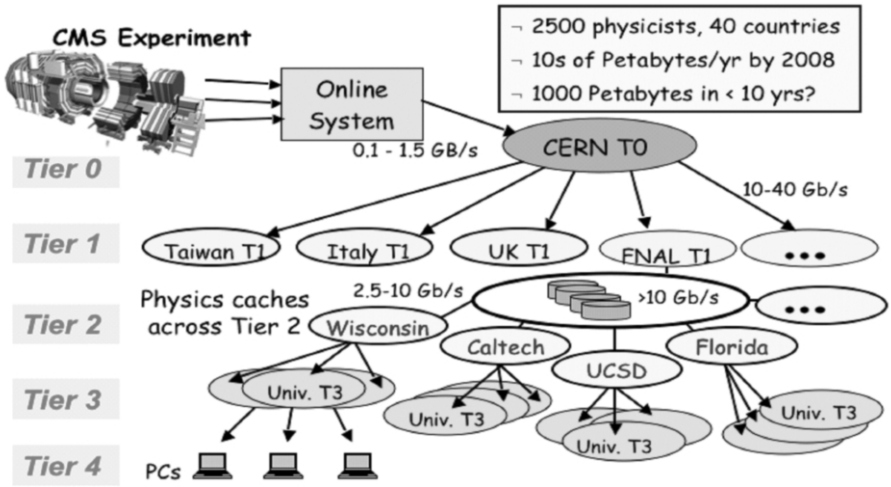


Figure 4. A refined view of the LHC Data Grid Hierarchy, developed in the DISUN project, where operations of the Tier-2 centers and the U.S. Tier-1 center are integrated through network connections with typical speeds in the 10 Gbps range.

Lab, face similar challenges. BaBar, for example, has already accumulated datasets approaching a petabyte.

The HEP problems are among the most data-intensive known. Hundreds to thousands of scientist-developers around the world continually develop software to better select candidate physics signals from particle accelerator experiments such as CMS, better calibrate the detector and better reconstruct the quantities of interest (energies and decay vertices of particles such as electrons, photons and muons, as well as jets of particles from quarks and gluons). These are the basic experimental results that are used to compare theory and experiment. The globally distributed ensemble of computing and data facilities (e.g., see Fig. 4), while large by any standard, is less than the physicists require to do their work in an unbridled way. There is thus a need, and a drive, to solve the problem of managing global resources in an optimal way in order to maximize the potential of the major experiments to produce breakthrough discoveries.

Collaborations on this global scale would not have been attempted if the physicists could not assume the existence of reliable, high capacity, feature-rich networks: to interconnect the physics groups throughout the lifecycle of the experiment, and to make possible the construction of Data Grids capable of providing access, processing and analysis of massive datasets. These datasets will increase in size from petabytes to exabytes (10^{18} bytes) within the next decade. Equally as important is highly capable middleware (the Grid data management and underlying resource access and management services) that is used to facilitate the management of world wide computing and data resources that must all be brought to bear on the data analysis problem of HEP [6].

Tiered Model of Regional Computing and Analysis Centers

Building on developments in the early HEP grid projects (PPDG and GriPhyN/iVDGL in the US, and the EU DataGrid), the LHC experiments have adopted the Data Grid Hierarchy of four “Tiers” of globally distributed computing and storage resources. Data at the experiment are stored at the rate of 200–1500 Mbytes/sec throughout the year,

resulting in many Petabytes per year of stored and processed binary data that are accessed and processed repeatedly by worldwide collaborators.

Referring to Fig. 4, processing and analyzing the data requires the coordinated use of the entire ensemble of Tier-N facilities. The relatively few large Tier-0 and Tier-1 facilities are best suited for the high priority large-scale tasks of systematic data processing, archiving and distribution, and data curation. Moving down the hierarchy to the smaller and more numerous Tier-2 and Tier-3 facilities, individuals and small groups have greater control over how these resources are allocated to small and medium-sized tasks of special interest to them. The Tier-2s, which comprise an estimated 40% of the overall CPU resources, are also foreseen to be the source of most of the simulated data and where most of the later-stage data analysis will take place.

The basic effectiveness of the grid hierarchy concept in a large-scale production setting is being shown clearly in the large-scale grid-based production operations of the LHC experiments, in partnership with the major grid projects OSG and EGEE [19]. The increasing scale and efficiency of these operations supporting the LHC and other major HEP experiments, as well as other science communities, has been accompanied by an increasing efficiency and scale of network usage.

While the top-down picture of the hierarchical computing model and its use in the LHC service challenges has been relatively simple until now, effective use of the compute and storage resources at Tier-2s would benefit greatly from more opportunistic data distribution and local data access. There will therefore be a tendency towards more dynamic data flow among the Tiers, as a growing number of physics groups learn to use the production-oriented and standalone tools effectively. In the longer run, when the community of thousands of physicists will share both local and more remote resources to analyze their data, dynamic and efficient use of the network would enable the community to balance its resource usage, and to make more effective use of local and regional resources where a group may have higher priority.

Refined View of the LHC Computing Model

At the start of LHC data-taking in 2007–2008, a typical Tier-2 site is expected to comprise of order 500–1000 kSi2000[2] of CPU power, and 100–300 TBytes of useable disk space for each experiment served.

Given the scale and nature of storage at the Tier-2s, none of the individual Tier-2 sites will have sufficient resources to host all the relevant data samples for its regional user community. Instead, there will be a need to dynamically move data and user applications among the collection of Tier-2 sites and the corresponding Tier-1 center (for example, the U.S. Tier-2s and U.S. Tier-1s at Fermilab or Brookhaven) in order to optimally exploit the physics potential of the experiment. Accordingly, there will be a corresponding need for the Tier-2 centers to be connected by high bandwidth networks.

Responding to this vision and the corresponding needs, four of the U.S. CMS Tier-2s have initiated the DISUN project, illustrated in Fig. 4. The physics data caches depicted at the center of the ring in the figure are distributed across Tier-2 sites, and are made available to scientists as a managed and high-throughput data resource supported by high throughput data transport services which are currently under development. It is also important to note that the diagram is schematic. The European Tier-2s are connected via the GÉANT2 network infrastructure, while the US Tier 1s and Tier 2s are

[2] CINT2000 is a measure of compute-intensive integer performance; kSi2000 is units of a thousand times the CINT2000 metric. An Intel P4 Xeon at 2.8GHz is approximately 1 kSI2000. See www.spec.org.

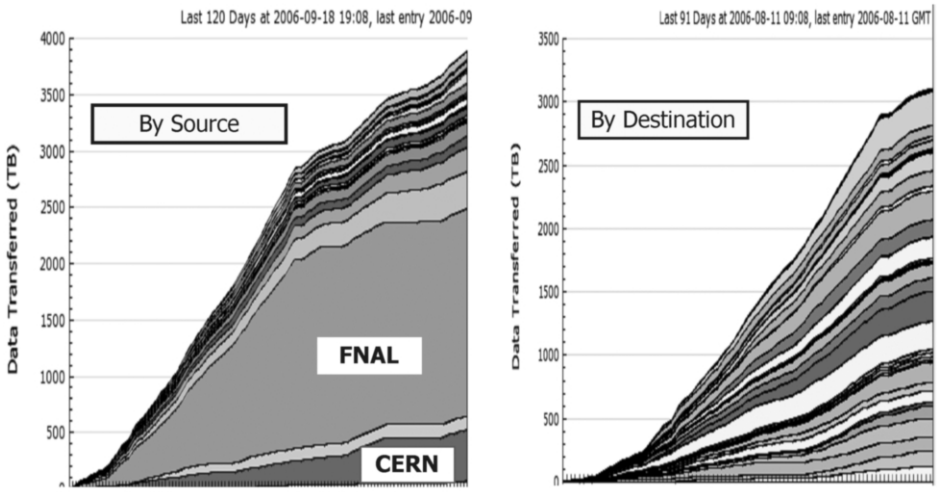


Figure 5. Data transfers by the CMS PhEDEx application. The graphs illustrate one of the LHC "Service Challenges" – application, site, and network readiness exercises – during 2006. In this case 1–2 petabytes/month data movement operated for 5 months. (Courtesy the CMS collaboration. See http://cmsdoc.cern.ch/cms/aprom/phedex/.)

interconnected via high-bandwidth peerings between ESnet and Internet2 at the major points of presence in Chicago (StarLight) and New York (MANLAN).

Nature of the Distributed Data Management and Analysis Systems

The LHC data management system has several characteristics that result in requirements for the network and its services.

- The systems are widely distributed – typically spread over continental or inter-continental distances. The systems are data intensive and high-performance, typically moving terabytes a day for months at a time (see Fig. 5).
- The system are high duty-cycle, operating most of the day for months at a time in order to meet the requirements for data movement.
- Such systems clearly depend on network performance and availability, but these characteristics cannot be taken for granted, even in well-run networks, when the multi-domain network path is considered. In fact, they cannot be taken for granted even within a single well-run, high-capacity network.
- The applications must be able to get guarantees from the network that there is adequate bandwidth to accomplish the task at hand. The applications must be able to get information from the network that allows graceful failure and auto-recovery and adaptation to unexpected network conditions that are short of outright failure (which is much more common than complete failure).

In other words, the network has to behave like a service that provides guarantees and information to support recovery when the guarantees are not met. The application then must be capable of using such information to implement dynamic reconfiguration strategies and so on.

As more experience is gained with the current generation of applications and prototype network services, several things are becoming clear. One is that the network has inadequate tools to monitor the new services like virtual circuits ("VC") and report back to the application in sufficient detail for the application to respond in an intelligent way. Another is that because VC services are relatively coarse-grained with respect to applications (VCs are typically set up between sites at this point), the application will have to share the bandwidth of a VC.

2.3. Network Requirements from Data and Collaboration Characteristics of DOE Office of Science Instruments, Facilities, and Science Practice

There are some 20 major instruments and facilities currently operated or being built by SC, plus the LHC (CERN, Switzerland) and ITER (France). To date, ESnet has characterized 14 of these for their future requirements. Facilities such as DOE's big accelerators (RHIC at Brookhaven, SNS at Oakridge) and supercomputer centers (NERSC at Lawrence Berkeley, NLCF at Oak Ridge, and ALCF at Argonne), as well as the LHC at CERN, are typical of the hardware infrastructure of the science supported by the Office of Science. These facilities generate four types of network requirements: bandwidth, connectivity and geographic footprint, reliability, and network services.

In order to determine the requirements of SC science based on how the process of conducting scientific research will change, a set of case studies were developed in which the science communities were asked to describe how they expected to have to be doing their science in five and ten years in order to make significant progress. Computer scientists then worked with the scientists to translate the new processes into network requirements – in particular those related to collaboration, data sharing and remote analysis, remote instrument control, and large-scale simulations coupled with each other and/or with external sources of data (e.g., operating instruments) [2]. Bandwidth needs are determined by the quantity of data produced and the need to move the data for remote analysis. Connectivity and geographic footprint are determined by the location of the instruments and facilities, and the locations of the associated collaborative community, including remote and/or distributed computing and storage used in the analysis systems. These locations also establish requirements for connectivity to the network infrastructure that supports the collaborators (e.g., ESnet connectivity to Internet2 and the US regional R&E networks, and GÉANT and the European national R&E networks – the NRENs).

The reliability requirements are driven by how closely coupled the facility is with remote resources. For example, off-line data analysis – where an experiment runs and generates data and the data is analyzed after the fact – may be tolerant of some level of network outages. On the other hand, when remote operation or analysis must occur within the operating cycle time of an experiment ("on-line" analysis, e.g., in magnetic fusion experiments), or when other critical components depend on the connection (e.g., a distributed file system between supercomputer centers), then very little network downtime is acceptable. The reliability issue is critical and drives much of the design of the network. Many scientific facilities in which DOE has invested hundreds of millions to billions of dollars, together with their large associated science communities, are heavily dependent on networking. Not surprisingly, when the experiments of these facilities depend on the network, then these facilities and scientists demand that the network provide very high availability (99.99+%), in addition to very high bandwidth.

The fourth requirement is in the area of types of service. In the past, networks typically provided a single network service – best-effort delivery of data packets[3] – on which are built all of today's higher-level applications (FTP, email, Web, socket libraries for application-to-application communication, etc.), and best-effort IP multicast (where a single outgoing packet is, sometimes unreliably, delivered to multiple receivers). In considering future uses of the network by the science community, several other network services have been identified as requirements, including bandwidth guarantees,[4] traffic isolation,[5] and reliable multicast.

Bandwidth guarantees are typically needed for on-line analysis, which always involves time constraints. Another type of application requiring bandwidth guarantees is distributed workflow systems such as those used by high energy physics data analysis. The inability of one element (computer) in the workflow system to adequately communicate data to another will ripple through the entire workflow environment, slowing down other participating systems as they wait for required intermediate results, thus reducing the overall effectiveness of the entire system.

Traffic isolation is required because today's primary transport mechanism – TCP – is not ideal for transporting large amounts of data across large (e.g., intercontinental) distances. There are protocols better suited to this task, but these protocols are not compatible with the fair-sharing of TCP transport in a best-effort network, and are thus typically penalized by the network in ways that reduce their effectiveness. A service that can isolate the bulk data transport protocols from best-effort traffic is needed to address this problem.

Reliable multicast is a service that, while not entirely new, must be enhanced to increase its effectiveness. Multicast provides for delivering a single data stream to multiple destinations without having to replicate the entire stream at the source, as is the

[3] Packet management by IP networks is not deterministic, but rather statistical. That is, the IP packets that make up, e.g., a TCP stream are injected into the network from many computers that are all connected to a single router – e.g. a typical large SC Lab will have many internal "subnets" all of which connect through different interfaces to a single site gateway router that provides connectivity to the outside world. The packets are queued in the router in whatever order they reach the routing processor (also called the forwarding processor). The packets in the queue waiting to be forwarded to their next-hop destination are intermixed indiscriminately by virtue of being queued immediately from several different input connections. As long as the queue does not overflow this is not an issue (in fact it is the norm) since every packet is routed through the network independently of every other packet. If the packets come into a router through several interfaces and they are all processed out through a single interface – as is typical, e.g., for a site gateway router that has several connections on the site side and a single connection on the Wide Area Network side – then it is possible for the forwarding processor to fall behind. This can happen either because the forwarding processor is not fast enough to keep up with the routing (which is rare in modern routers) or because the aggregate input traffic bandwidth exceeds the bandwidth of the single output interface (a circumstance that, in principle, is easily realized). When this happens the input queue for the forwarding engine will fill and "overflow" – this is called network congestion. The overflow process is a random discard of the incoming packets, and the overall effect is that there is no guarantee that a packet sent to a router is forwarded on to its next hop toward its destination – packet forwarding is a "best-effort" process. (Users typically see congestion as a slowdown in the network – they do not see the packet loss directly because most applications use TCP as a reliable transport protocol. TCP uses IP packets to move data through the network and it detects packet loss and automatically resends the lost IP packets in order to ensure reliable data delivery.)

[4] Bandwidth guarantees are provided in IP networks by doing two things: First, the packets in a bandwidth-guaranteed connection are marked as high priority and are forwarded ahead of any waiting best-effort packet. Second, the bandwidth-guaranteed connections are managed so that, in aggregate, they never exceed the available bandwidth anywhere in the path to their destination. This entails limiting the input bandwidth of a bandwidth-guaranteed connection to an agreed upon value, and then by limiting the number of such connections so as not to exceed the available bandwidth along the path.

[5] Traffic isolation is provided in a way similar to bandwidth guarantees in that the packets are queued and forwarded in such a way that they do not interact with other classes of traffic such as best-effort.

case, e.g., when using a separate TCP-based connection from the source to each receiver. This is important when the data to be delivered to multiple sites is too voluminous to be replicated at the source and sent to each receiving site individually. Today, IP multicast provides this capability in a fragile and limited way (IP multicast does not provide reliable delivery as TCP-based transport does). New services may be required to support reliable and robust multicast.

In the case studies that have been done to date [5], one or more major SC facilities have identified a requirement for each of these network capabilities.

The case studies of [2,4], and [5] were picked both to get a good cross-section of SC science and to provide realistic predictions based on highly probable changes in the scientific process in the future. The case studies were conducted over several years and included the following Office of Science programs and associated facilities: Magnetic Fusion Energy, NERSC, ACLF, NLCF, Nuclear Physics (RHIC), Spallation Neutron Source, Advanced Light Source, Bioinformatics, Chemistry/Combustion, Climate Science, and High Energy Physics (LHC).

Summary of the Conclusions of the Case Studies

There is a high level of correlation between network requirements for large and small-scale science – the primary difference being bandwidth – and so meeting the requirements of the large-scale stakeholders will generally provide for the requirements of the smaller ones, provided the required services set is the same.

Some of the non-bandwidth findings from the case studies included:

- The geographic extent and size of the user base of scientific collaboration is continuously expanding. As noted, DOE US and international collaborators rely on ESnet to reach DOE facilities, and DOE scientists rely on ESnet to reach non-DOE facilities nationally and internationally (e.g., LHC, ITER). Therefore, close collaboration with other networks is essential in order to provide high-quality end-to-end service, diagnostic transparency, etc.
- Robustness and stability (network reliability) are essential. Large-scale investment in science facilities and experiments makes network failure unacceptable when the experiments depend on the network.
- Science requires several advanced network services for different purposes. There are requirements for predictable latency and quality of service guarantees to support remote real-time instrument control, computational steering, and interactive visualization. Bandwidth guarantees and traffic isolation are needed for large data transfers (potentially using TCP-unfriendly protocols), and network support for deadline scheduling of data transfers.

The aggregation of requirements from the 14 case studies (see [5]) results in:

- Reliability
  - The Fusion requirements of 1 minute of down time during an experiment that runs 8–16 hours a day, 5–7 days a week, implies a network availability of 99.999%. LHC data transfers can only tolerate a small number of hours of outage in streams that operate continuously for 9 months per year, otherwise the analysis of the data coming from the LHC will fall too far behind to ever catch up. This implies a network availability of 99.95%.

  - These needs result in a requirement for redundancy (which is the only practical way to achieve this level of reliability) both for site connectivity and within ESnet.
- Connectivity
  - The geographic reach of the network must be equivalent to that of the scientific collaboration. Multiple peerings with the other major R&E networks are needed to add reliability and bandwidth for inter-domain connectivity. This is critical both within the US and internationally.
- Bandwidth
  - A bandwidth of 10 Gb/s site-to-site connectivity is needed now, and 100 Gb/s will be needed by 2010. Multiple 10 Gb/s peerings (interconnections) with the major R&E networks will be needed for data transfers. The network must have the ability to easily deploy additional 10 Gb/s circuits and peerings as needed by new science projects.

Bandwidth and service guarantees are needed end-to-end, so all R&E networks must interoperate as one seamless fabric. Flexible rate bandwidth guarantees are needed – that is, a project must be able to ask for the amount of bandwidth that it needs and not be forced to use more or less.

The case studies include both quantitative and qualitative requirements.

2.4. Requirements from Observing Traffic Patterns

From the analysis of historical traffic patterns, several clear trends emerge that result in requirements for the evolution of the network so it can handle the projected traffic load.

The first, and most obvious, pattern is the exponential growth of the total traffic handled by ESnet (Figs 6 and 7). This traffic trend represents a 10× increase every 47 months on average since 1990 (Fig. 7). ESnet traffic just passed the 1 petabyte per month level with about 1.5 Gb/s average, steady-state load on the New York-Chicago-San Francisco path. If this trend continues (and all indications are that it will accelerate), the network must be provisioned to handle an average of 15 Gb/s in four years. This implies a minimum backbone bandwidth of 20 Gb/s, because the network peak capacity must be at least 40% higher than the average load in order for today's protocols to function properly with bursty traffic (which is the norm). In addition, the current traffic trend suggests that 200 Gb/s of core network bandwidth will be required in eight years. This can only be achieved within a reasonable budget by using a network architecture and implementation approach that allows for cost-effective scaling of hub-to-hub circuit bandwidth.

The second major change in traffic is the result of a dramatic increase in the use of parallel file mover applications (e.g., GridFTP). This has resulted in the most profound change in traffic patterns in the history of ESnet. Over the past two years, this has resulted in a change from the historical trend where the peak system-to-system ("workflow") bandwidth of the largest network users increased along with the increases in total network traffic, to a situation where the peak bandwidth of the largest user systems is coming down, and the number of flows that they generate is going up, while the total traffic continues to increase exponentially. This reduction in peak workflow bandwidth, together with an overall increase in bandwidth, is the result of the decomposition of single large flows into many smaller parallel flows. In other words, the

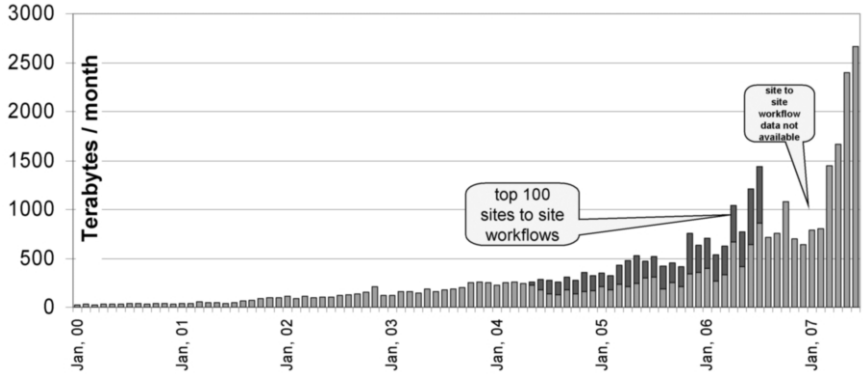


Figure 6. Total ESnet traffic by month, 2000–2007. The segmented bars from mid-2004 on show that fraction of the total traffic in the top 1000 data flows (which are from large-scale science facilities). (There are typically several billion flows per month in total, most of which are minuscule compared to the top 1000 flows.)

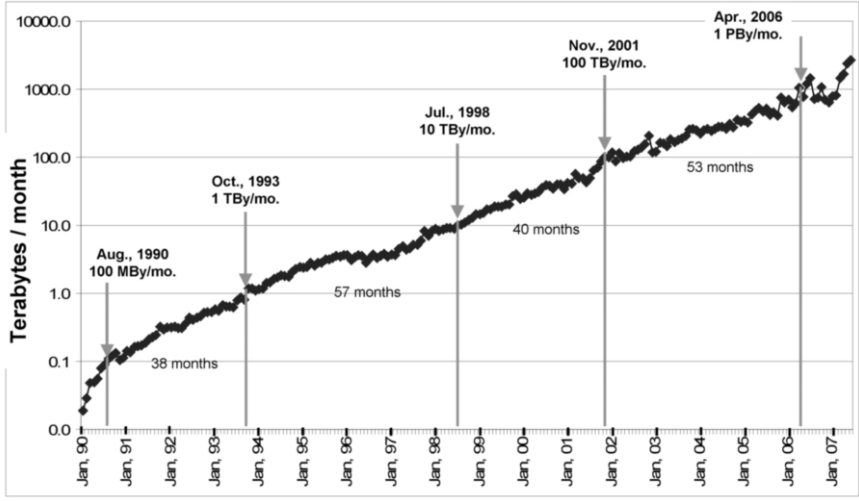


Figure 7. Log plot of ESnet traffic since 1990.

same types of changes that happened in computational algorithms as parallel computing systems became prevalent are now happening in data movement – that is, parallel I/O channels operating across the network. This is illustrated in Fig. 8, where the top 100 host-to-host data transfers, in one month averages, for a sampling of months over the past 18 months, are represented in the bar charts labeled "Host to Host Traffic." (The "stair-step" appearance arises from groups of associated parallel file movers that move approximately the same amount of data while operating.) Next to these graphs is the total network traffic for that month, segmented as in Fig. 6.

The third clear traffic trend is that over the past two years the impact of the top few hundred workflows – there are of order 6×10^9 flows per month in total – has grown from negligible before mid-2004 to more than 50% of all traffic in ESnet by mid-2006!

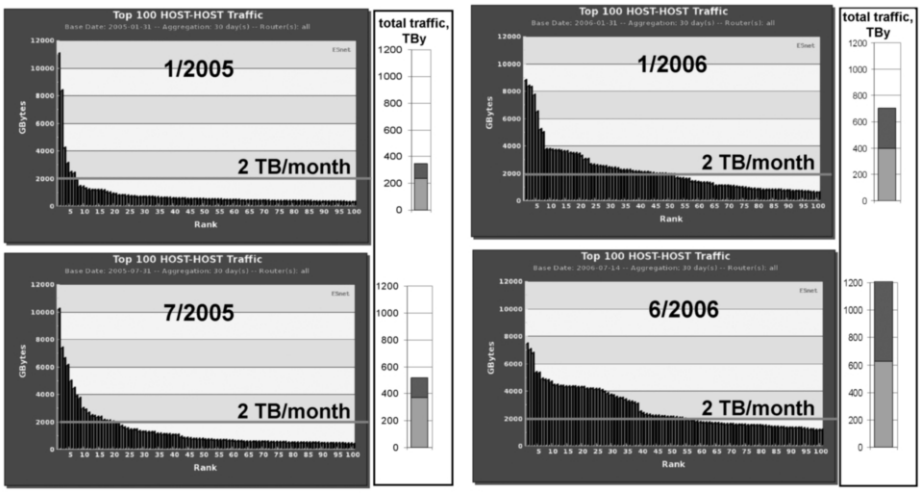


Figure 8. ESnet's traffic patterns are evolving due to increasing use of parallel file movers.

This is illustrated in Fig. 6, where the top part of the traffic bars shows the portion of the total generated by the top 100 hosts.

The fourth significant pattern comes from looking at the source and destination locations of the top data transfer systems – an examination that shows two things. First is that the vast majority of the transfers can easily be identified as science traffic since the transfers are between two scientific institutions with systems that are named in ways that reflect the name of the science group. Second, for the past several years the majority of the large data transfers have been between institutions in the US and Europe and Japan, reflecting the strongly international character of large science collaborations organized around large scientific instruments (Fig. 9).

Finally, Fig. 9 – only somewhat jokingly referred to as the "onslaught of the LHC" – also illustrates the limitation of using traffic trends alone to predict the future network needs of science. No traffic observations could have predicted the upsurge in LHC data movement, both from CERN to the SC Labs and from the SC Labs to US universities. Obviously traffic trend analysis cannot predict the start of new science projects.

2.5. Network Requirements Summary

The combination of the case studies and the traffic pattern trends adds quantitative aspects to the general requirements that were identified early in this paper.

The aggregate network capacity must reach 100–200 Gb/s in the five- to seven-year time frame. Network reliability must increase from the historical 99.9% to 99.99% to something more like 99.99% to 99.999% availability to the end site. The peerings – external network interconnections between national R&E and international R&E networks and ESnet – must increase both in bandwidth and reliability in a similar fashion. In addition, several specific new network services related to bandwidth guarantees must be introduced into the production network.

A general requirement is that there must be flexibility in provisioning the network capacity. The location of the greatest need for bandwidth within the network will

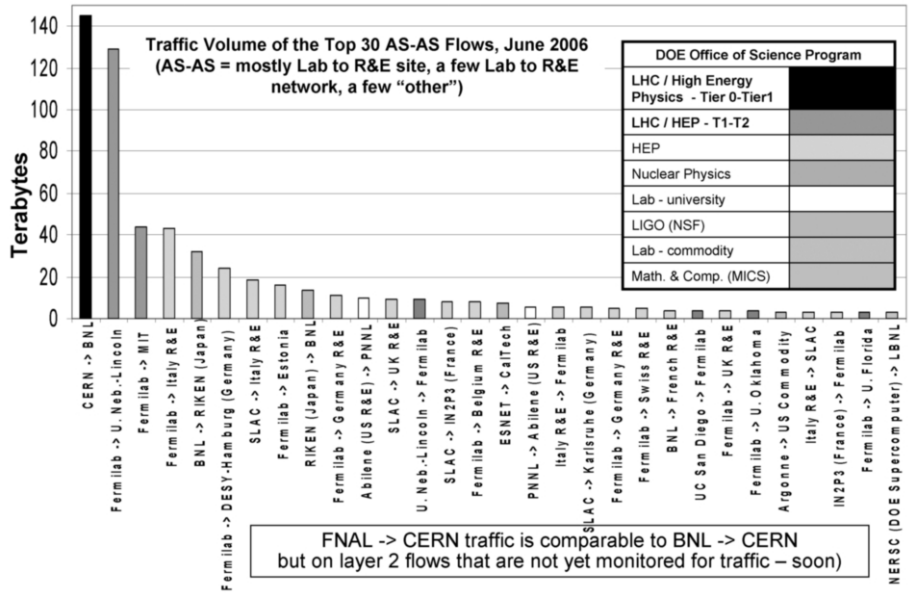


Figure 9. Traffic patterns due to new uses of the network by the LHC. LHC to BNL is the No. 1 traffic generator; FNAL to and from US universities accounts for Nos. 2, 3, 13, 23, 24, and 28.

change over time, and the budgetary resources available for the network may also change. It must be possible add and move hub-to-hub capacity as needed and to deploy new capacity on a schedule determined by science needs and funding availability.

3. Enabling Future Science: ESnet's Evolution over the Next 10 Years

Based both on the projections of the science programs and the changes in observed network traffic and patterns over the past few years, it is clear that the network must evolve substantially in order to meet the needs of DOE's Office of Science.

The current trend in traffic patterns – the large-scale science projects giving rise to the top 100 data flows that represent about 1/2 of all network traffic – will continue to evolve. As the LHC experiments ramp up in 2006-07, the data to the Tier-1 centers (FNAL and BNL) will increase 200–2000 times. A comparable amount of data will flow out of the Tier-1 centers to the Tier-2 centers (U.S. universities) for data analysis. The DOE National Leadership Class Facility supercomputer at ORNL anticipates a new model of computing in which simulation tasks are distributed between the central facility and a collection of remote "end stations" that will generate substantial network traffic. As climate models achieve the sophistication and accuracy anticipated in the next few years, the amount of climate data that will move into and out of the NERSC center will increase dramatically (they are already in the top 100 workflows) Similarly, the experiment facilities at the new Spallation Neutron Source and Magnetic Fusion Energy facilities will start using the network in ways that require fairly high bandwidth with guaranteed quality of service.

This evolution in traffic patterns and volume will result in the top 100–1000 flows accounting for a very large fraction of all the traffic in the network, even as total ESnet

traffic volume grows: The large-scale science data flows will overwhelm everything else on the network.

By 2009/2010 the current, few gigabits/sec of average traffic on the backbone will increase to 40 Gb/s (LHC traffic) and then increase to probably double that amount as the other science disciplines move into a collaborative production simulation and data analysis mode on a scale similar to the LHC. This will get the backbone traffic to 100 Gb/s in 2010–2012 as predicted by the science requirements analysis three years ago.

The old ESnet hub and spoke architecture (through 2004) would not have let ESnet meet these new requirements. The current core ring cannot be scaled to handle the anticipated large science data flows at affordable cost. Point-to-point, commercial telecom tail circuits to sites are neither reliable nor scalable to the required bandwidth.

3.1. ESnet4: A New Architecture to Meet the Science Requirements

In order to accommodate this growth, and the change in the types of traffic, the architecture of the network must change. The general requirements for the new architecture are that it provide:

- High-speed, scalable, and reliable production IP networking, connectivity for University and international collaboration, highly reliable site connectivity to support Lab operations as well as science, and Global Internet connectivity.
- Support for the high bandwidth data flows of large-scale science including scalable, reliable, and very high-speed network connectivity to DOE Labs.
- Dynamically provisioned, virtual circuits with guaranteed quality of service (e.g. for dedicated bandwidth and for traffic isolation).

In order to meet these requirements, the capacity and connectivity of the network must increase to include fully redundant connectivity for every site, high-speed access to the core for every site (at least 20 Gb/s, generally, and 40–100 Gb/s for some sites) and a 100 Gb/s national core/backbone bandwidth by 2009/2010 in two independent backbones.

The strategy for the next-generation ESnet is based on a set of architectural principles that lead to four major network elements and a new network service for managing large data flows.

The architectural principles are:

- Use ring topologies for path redundancy in every part of the network – not just in the WAN core.
- Provide multiple, independent connections everywhere to guard against hardware and fiber failures.
- Provision one core network – the IP network – specialized for handling the huge number (3×10^9/mo.) of small data flows (hundreds to thousands of bytes each) of the general IP traffic.
- Provision a second core network – the Science Data Network (SDN) – specialized for the relatively small number (hundreds to thousands) of massive data flows (gigabytes to terabytes each) of large-scale science (which by volume already accounts for 50% of all ESnet traffic and will completely dominate it in the near future).

These architecture principles lead to four major elements for building the new network:

- A high-reliability IP core network based on high-speed, highly capable IP routers to support:
  - Internet access for both science and lab operational traffic, and some backup for the science data carried by SDN
  - science collaboration services
  - peering with all of the networks needed for reliable access to the global Internet.
- A Science Data Network core network based on Ethernet switches that support Multi-Protocol Label Switching (MPLS) and/or layer 1[6] (optical) switches for:
  - multiple 10 Gb/s circuits with a rich topology for very high total bandwidth to support large-scale science traffic and for the redundancy needed to high reliability
  - dynamically provisioned, guaranteed bandwidth circuits to manage large, high-speed science data flows
  - dynamic sharing of some optical paths with the R&E community for managing peak traffic situations and for providing specialized services such as all-optical, end-to-end paths for uses that do not yet have encapsulation interfaces (e.g. Infiniband)
  - an alternate path for production IP traffic.
- Metropolitan Area Network (MAN) rings connecting labs to the core(s) to provide:
  - more reliable (ring) and higher bandwidth (multiple 10 Gb/s circuits) site-to-core connectivity
  - support for both production IP and large-scale science traffic
  - multiple connections between the Science Data Network core, the IP core, and the sites.
- Loops off the core rings to provide for dual connections to remote sites where MANs are not practical.

These elements are structured to provide a network with fully redundant paths for all of the SC Labs. The IP and SDN cores are independent of each other and both are ring-structured for resiliency. These two national cores are interconnected at several locations with ring-structured metropolitan area networks that also incorporate the DOE Labs into the ring. This will eliminate all single points of failure except where multiple fibers may be in the same conduit (as is frequently the case between metropolitan area points of presence and the physical sites). In the places where metropolitan rings are not practical (e.g. the geographically isolated Labs) resiliency is obtained with dual connections to one of the core rings (see Fig. 10).

[6] The "layer" term refers to the Open Systems Interconnect (OSI) standard model. Very briefly, layer 1 refers to the sending and receiving bits at the optical or electrical interface. Layer 2 refers to how a computer gets access to a network – e.g. via an Ethernet interface. Layer 3 refers to routing and switching (e.g. IP routers) and layer 4 refers to data transport (e.g. TCP). The OSI model does not map perfectly onto the IP model, but the terms are used anyway. Likewise referring to an Ethernet switch as a "layer 2" device and an IP router as a "layer 3" is not strictly accurate since almost all modern Ethernet switches can do some IP routing and almost all IP routers can so some Ethernet switching. Again, however, the terms are used anyway.

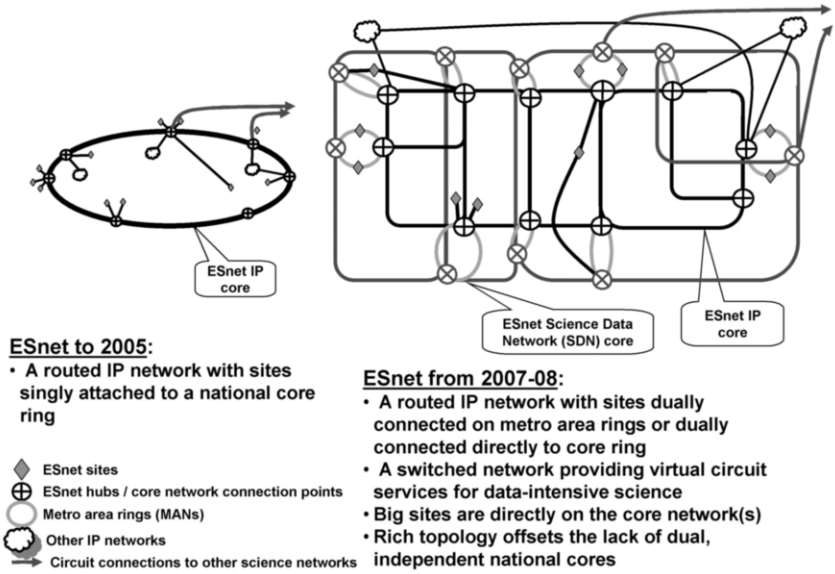


Figure 10. The evolution of the ESnet architecture.

The theoretical advantages of this architecture are clear but it must also be practical to realize in an implementation. That is, how does ESnet get to the 100 Gb/s multiple backbones and the 20–40 Gb/s redundant site connectivity that is needed by the SC community in the 3–5 yr time frame?

3.2. Building ESnet4

Internet2 – the network that serves the US R&E community–has partnered with Level 3 Communications Co. and Infinera Corp. to build a dedicated optical fiber infrastructure with a national footprint and a rich topology – the "Internet2 Network."

The fiber will be provisioned with Infinera Dense Wave Division Multiplexing equipment that uses an advanced, integrated optical-electrical design. Level 3 will maintain the fiber and the DWDM equipment as part of its commercial network–a very important consideration for reliability. The DWDM equipment will initially be provisioned to provide10 optical circuits (lambdas or waves) across the entire fiber footprint (40 waves is the current configuration capacity, 80 is maximum.)

ESnet has partnered with Internet2 to:

- Share the optical infrastructure
- Develop new circuit-oriented network services
- Explore mechanisms that could be used for the ESnet Network Operations Center (NOC) and the Internet2/Indiana University NOC to back each other up for disaster recovery purposes.

ESnet will build its next generation IP network and its new circuit-oriented Science Data Network primarily on Internet2 optical circuits that are dedicated to ESnet, together with a few from National Lambda Rail and others. ESnet will provision and operate its own routing and switching hardware that is installed in various commercial

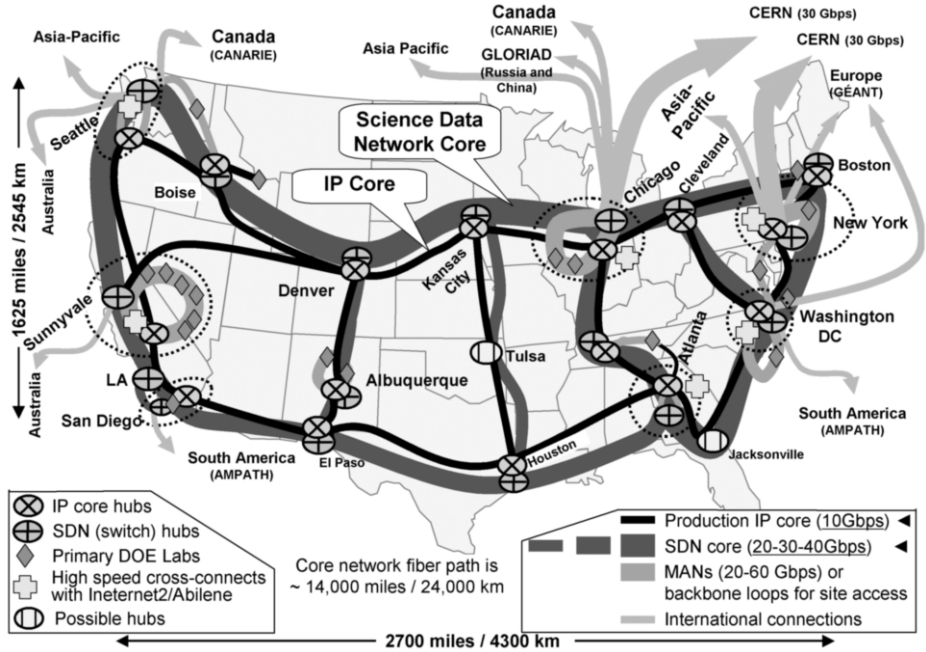


Figure 11. ESnet4, 2012 configuration. The next generation of optical DWDM equipment and network switches and routers is expected to be in place by 2010–2011 to provide 10X over the current per-circuit bandwidth – that is 100 Gb/s per circuit. The core networks will grow to 40–50 Gbps in 2009–2010 and, with new technology, to 400–500 Gbps in 2011–2012.

telecom hubs around the country, as it has done for the past 20 years. ESnet's peering relationships with the commercial Internet, various US research and education networks, and numerous international networks will continue and evolve as they have for the past 20 years.

ESnet4 will also involve an expansion of the multi-10Gb/s Metropolitan Area Rings in the San Francisco Bay Area, Chicago, Long Island, Newport News (VA/Washington, DC area), and Atlanta to provide multiple, independent connections for ESnet sites to the ESnet core network. (Building the Metropolitan Area Networks that get the Labs to the ESnet cores is a mixed bag and somewhat opportunistic – a combination of R&E networks, dark fiber networks, and commercial managed lambda circuits are used.) In fact, in the new architecture all of the big SC Labs are effectively connected directly to both the IP and SDN core networks.

3.3. New Network Services

New network services are also critical for ESnet to meet the needs of large-scale science (see [2–4] and [5]).

Dynamically provisioned virtual circuits that provide traffic isolation are needed to enable the use of non-standard transport mechanisms that cannot co-exist with TCP-based transport and provide guaranteed bandwidth.

Guaranteed bandwidth was identified as very important in three specific situations.

The first situation is that it is the only way that we currently have to address deadline scheduling – e.g. where fixed amounts of data have to reach sites on a fixed schedule in order that the processing does not fall so far behind that it could never catch up. This is very important for certain experiment's data analysis.

The second situation is where remote computing elements are involved in control of real-time experiments. Two examples of this were cited in the applications requirements workshop [2] – one from magnetic fusion experiments and the other from the Spallation Neutron Source. The magnetic fusion situation is that theories are tested with experiments in Tokamak fusion reactors. The experiments involve changing the many parameters by which the reactor can operate and then triggering plasma generation. The "shot" (experiment) lasts a few 10s of milliseconds and generates hundreds of megabytes of data. The device takes about 20 minutes to cycle for the next shot. In that 20 minutes the data must be distributed to the remote collaborators, analyzed, and the results of the analysis fed back to the reactor in order to set up the next experiment (shot). In order to have enough time to analyze the data and use the parameters to set up the next experiment, 200–500 Mb/s of bandwidth must be guaranteed for 2–5 minutes to transmit the data and leave enough time to do that analysis. The situation with the SNS is similar.

The third situation is when Grid based analysis systems consist of hundreds of clusters at dozens of universities that must operate under the control of a workflow manager that choreographs complex workflows. This requires network quality of service to ensure a steady flow of data and intermediate results among the systems. Without this, systems with many inter-dependencies could stop and start, causing interruptions that would propagate throughout the entire collection of systems. This would create an unstable and inefficient production environment that would reduce the overall throughput necessary to keep up with the steady generation of data by the experiment. (This is of particular concern with the huge amount of data coming out of the LHC experiments.)

In addition to virtual circuits, another new network service that is essential is an end-to-end monitoring service. As cross-domain virtual circuit services start to be deployed, monitoring is seen as a critical service that is needed both for network operators and users.

4. Development and Deployment of Service-Oriented Communication Services

DOE SC has funded the OSCARS (On-demand Secure Circuits and Advance Reservation System) project to develop and deploy the various technologies that provide dynamically provisioned circuits and quality-of-service (QoS) that can be integrated into a production network environment. Such "circuits" are called "virtual circuits" (VCs) because that are defined in software and thus are mutable (as opposed to hardware established circuits).

The elements of this system (illustrated in Fig. 12) are the

- Web-Based User Interface (WBUI) that will prompt the user for a username/password and forward it to the AAAS (Authentication, Authorization, and Auditing Subsystem)
- Authentication, Authorization, and Auditing Subsystem that will authenticate users, handle access authorization, enforce policy, and generate usage records

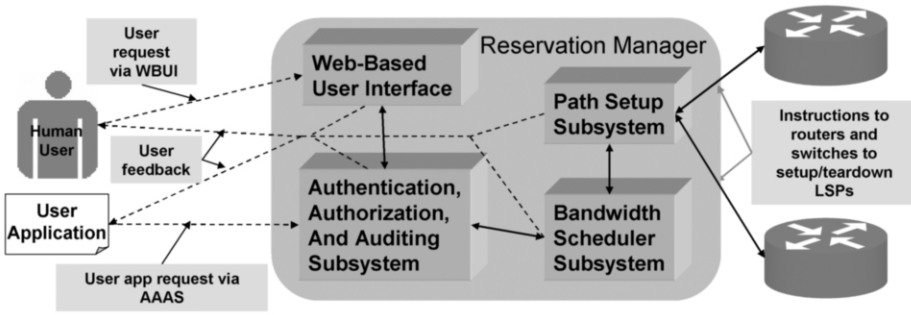


Figure 12. Architecture of the OSCARS Virtual Circuit management system.

- Bandwidth Scheduler Subsystem (BSS) that will track reservations and map the state of the network (present and future)
- Path Setup Subsystem (PSS) that will setup and teardown the on-demand paths (VCs).

The end-to-end provisioning of VCs will initially be provided by a combination of Ethernet switch management of optical channel circuits in the MANs and Ethernet VLANs managed as MPLS paths (Multi-Protocol Label Switching and Label Switched Paths – LSPs) in the SDN core and as MPLS VCs in the IP network.

There are two realms in which OSCARS must operate: 1) intra-domain – that is, to establish a schedulable, guaranteed bandwidth circuit service within the boundary of the ESnet network; 2) inter-domain – e.g. to provide end-to-end QoS between DOE Labs and US and European universities.

Setting up inter-domain guaranteed bandwidth circuits is not a trivial task. It typically involves the virtual circuit extending across five to seven autonomous networks: the lab/campus network at each end, the lab/campus service provider (e.g. ESnet, a US RON (Regional Optical Network), or a European NREN) and the US national or pan-European transit network (e.g. ESnet, Internet2, GÉANT) or SINet (Japan). Differences in network infrastructure (e.g. hardware, link capacity, etc.) must be addressed at the inter-domain boundary in order to provide consistent service characteristics (e.g. bandwidth, delay, and jitter) across domains, as must the issues of different policies, such as Acceptable Use Policies (AUPs), Service Level Agreements (SLAs) and security requirements. Nevertheless, inter-domain circuits are essential, especially between ESnet, Internet2, and GÉANT. (Note that OSCARS does not address the important issue of inter-domain brokering policies. Enforcement of such policies, however, are critical to the deployment of OSCARS as a production service. Collaborative work is being done with the GÉANT, Joint Research Activity 5 project to ensure a compatible authentication/authorization framework.)

In the absence of agreed upon standards for the inter-domain interface (called an "ENNI" – external network-network interface) the community is ensuring interoperability by collaboratively developing the software. This collaboration currently involves joint code development with the Internet2 BRUW project, and is working with HOPI (Internet2), TeraPaths (Brookhaven Lab), and DRAGON (an NSF-funded project) to ensure interoperability between each of these projects. OSCARS is also working with HOPI (Internet2), JRA5 (GÉANT's Joint Research Activity 5 project) to define an appropriate and interoperable AAI framework. OSCARS is working with DICE Control

Plane group to determine schemas and methods of distributing topology and reachability information, multi-domain scheduling, and inter-domain signaling (DICE=Internet2, ESnet, GÉANT, CANARIE/UCLP; see http://www.garr.it/dice/presentation.htm); and working with Tom Lehman (DRAGON), Nagi Rao (USN), Nasir Ghani (Tennessee Tech) on multi-level, multi-domain hybrid network performance measurements. A number of OSCARS circuits are currently being tested between various institutions.

For more information on the OSCARS implementation see http://www.es.net/oscars.

5. The Critical Role of Monitoring and Reporting

In order to build large-scale, widely distributed systems that operate reliably to perform complex data analysis (cf. LHC Case Study, above) or computational simulation tasks, the distributed applications and middleware must be able to learn, in real-time, about unexpected changes in the state of the communication between all of its components. Without this capability human users or system operators are left trying to intuit what has gone wrong. A problem that appears to come from one component may actually be an unreported communications problems from a very different part of the system. A reliable network monitoring service that describes the current state of application communications allows applications to adapt their behavior to changing circumstances, or at least to fail gracefully and accurately announce why it is failing.

An essential change in network services over the next five years will be to provide reliable, comprehensive, timely, and interpretable information about the state of all networks components in the end-to-end path in a manner that can be meaningfully interpreted and used by user-level applications. This ability must be accompanied by a corresponding capability in the applications and middleware to accept the communication services monitoring results and do something intelligent with those results. This may include adapting the functioning of the system to the changed/diminished communication service capability, graceful shutdown of the system, notifying the user what is happening (in terms that are useful to the users involved), and so on. The monitor results must be presented in a way that is meaningful to the user's view of the network.

Together with the new capabilities provided by virtual circuits, monitoring services that can report problems directly to the networked applications and users[7] will move network communications toward a managed service model more like the computing environment provides.

5.1. Background

All networks do extensive real-time monitoring which is used for a variety of uses. Short-term monitoring (on the order of minutes) is used for identification and debugging of problems in every element of the network – circuits, interfaces, switching and routing equipment, routing state (logical connectivity), and so on. This monitoring is primarily used to detect failure or failure onset through degraded performance or some aspect of the many network element health indicators. ESnet, for example, monitors almost 5,000 network element characteristics in real time in its national network. A commercial network monitoring system (Spectrum) is used to manage this information, generate operator alerts, and so on.

[7] We will user the term "user" to interchangeably mean an application agent or service or a human user.

Intermediate term (hours to days) interface traffic monitoring is done for capacity management: Hotspots can develop in the network due to changes in the user demand or capability, changes in network capacity (augments, or outages), or routing changes. It may be possible to address these hotspots by configuration (routing) changes, as networks are growing more densely meshed internally and more richly connected to each other. This sort of information could also trigger physical reconfiguration of parts of the network – typically by increasing interface bandwidth when possible.

Long term (months to years) traffic trend monitoring supports planning future network configurations, etc.: Traffic trends that show up over months or years (e.g. Figs 6–8 and 9) are essential in planning future architecture changes and major upgrades that will occur years in the future. These are one of the several metrics that drive the design of the next generation of the network.

Typically, detailed (minute-level granularity) network interface usage is available on-line for about a month and is then archived for future reference. Summary information (monitor data summarized at hourly, daily, or weekly granularity) is available online for several years. (ESnet, for example, monitors almost a thousand logical network interfaces on 64 routers and switches, and collects and archives about 325 GBy/month of monitor data.)

5.2. Network Monitoring System Design Goals

Detailed real-time network link state and performance data is routinely collected and archived in almost all production networks. However, what is of interest to the network operators is the behavior of specific router or switch interfaces and the link connecting them. Therefore the data is collected and data archives are organized and indexed in this fashion. Further, the form of this data is typically peculiar to each network, making the information almost useless to the user trying to see end-to-end behavior. In order to be useful to the user for end-to-end monitoring, the information must satisfy an additional set of requirements.

There must be tools to map the user view (as represented, e.g., by a traceroute of the application-application path) to the network view and then collect and map the corresponding network monitoring data back to the user view. That is, the tools must convert the user view to the physical path representation – the sets of interfaces and links that comprise the path at the physical level; extract the related data from the archive; map it back to the user view; and return the results to the user in a format that is standard across all networks.

Further, the entire end-to-end path must be included in the monitoring. In a typical R&E environment such paths involve five to six network domains: the site LAN, the regional or national network, a second national or a pan-national network, back into a regional or national network, and into the site LAN at the other end of the path. Each of these domains must provide the data for the segments of the user path that are part of that domain. This sort of cross-domain monitoring is critical both for high-performance applications that depend on widely distributed components and for network operators who are increasingly required to manage end-to-end paths.

A user should be able to be notified of service outages by subscribing to alerts for a given application path. Further, the report should provide information about the source of the outage – is it due to congestion (to which the user may be contributing), or link errors which is a network problem, or some other problem. Currently network

operators log planned outages in a calendar system and this system must also report future outages to the user.

Again, the problem with this from the user point of view is that the descriptions are given in terms of the physical topology of the network. To be useful to the user, physical topology must be mapped into user path descriptions and point failures must be reported in terms of their impact on the user path.

5.3. New Monitoring Services

PerfSONAR[8]

PerfSONAR is intended as a significant first step in cross-domain monitoring by both network operators and users.

Quoting from the perfSONAR Web site (www.perfsonar.net):

PerfSONAR has three contexts:

1. *perfSONAR is first a consortium of organizations who seek to build network performance middleware that is interoperable across multiple networks and useful for intra- and inter-network analysis. One of the main goals is to make it easier to solve end-to-end performance problems on paths crossing several networks.*
2. *perfSONAR is a protocol. It assumes a set of roles (the various service types), defines the protocol standard (syntax and semantics) by which they communicate, and allows anyone to write a service playing one of those roles. The protocol is based on SOAP XML messages and following the Open Grid Forum (OGF) Network Measurement Working Group (NM-WG).*
3. *perfSONAR is, finally, an example set of code (implementation of services) that attempts to implement an interoperable performance middleware framework. Those sets of code are developed by different partners. Some pieces of code are "more important" than others because their goal is to ensure interoperability between domains (e.g. the Lookup Service and the Authentication Service). Different subsets of code are important to each partner, with a great deal of overlap. The services developed acts as an intermediate layer, between the performance measurement tools and the diagnostic or visualization applications.*

Functionality: In order to satisfy the needs of the various communities of users of network data – the network operators and engineers, the network support staff at the institutions of the end users, and the end users both in the process of debugging the performance of a distributed application or as part of a service that reports network problems to an application resource manager – there are several aspects of network monitoring that must be addressed.

---

[8] This section draws on Hanemann, A., Boote, J. W., Boyd, E. L., Durand, J., Kudarimoti, L., Lapacz, R., Swany, D. M., Zurawski, J., Trocha, S., "PerfSONAR: A Service Oriented Architecture for Multi–Domain Network Monitoring", In "Proceedings of the Third International Conference on Service Oriented Computing", Springer Verlag, LNCS 3826, pp. 241–254, ACM Sigsoft and Sigweb, Amsterdam, The Netherlands, December, 2005 and Hanemann, A., Liakopoulos, A., Molina, M., Swany, D. M., "A Study on Network Performance Metrics and their Composition" TERENA Networking Conference 2006. – download; also appeared in special edition of Campus-Wide Information Systems (Volume 23 – 4 – 2006 – ISSN 1065-0741), Emerald Publishing Group Ltd. For more information see these and other papers at www.perfsonar.net.

There are three general categories of performance measurement data – active measurements, passive measurements, and network state variables (SNMP variables) – that can be thought of as data producers. From the network data user's point of view this data must be available in various ways and must have various services associated with it both to homogenize the information from different networks and to present the data in useful ways. Data should be provided as a data flow or via polling.

The analysis tools, threshold alarms, and visualization tools are data consumers that, in turn, need data that is already transformed in various ways. Therefore, between data producers and data consumers there may be a pipeline of aggregators, correlators, filters, and buffer services that can be regarded as data transformers and data archives.

Further, the services – the data producers, consumers, transformers, and archives – are all resources that need to be discovered and almost certainly used within an authentication and authorization framework that maintains the policy prescribed by the network operators that own the measurement data.

Architecture: A service oriented architecture (SOA) has been adopted by the community that consists of three layers and a collection of defined service functions (see Fig. 13).

- The *Measurement Point layer* is the lowest layer of the architecture. It collects network measurements, transforms the results into a standard format, and publishes the information to a Measurement Archive, or other service.
- The *Service layer* includes data management, manipulation, and transformation services and a collection of "housekeeping" services that provide standard authentication and authorization, service discovery, etc. The service layer is not a simple in-and-out layer, but contains pipeline or compound services like the Measurement Archive are both a service and a consumer of services.
- The *Interface layer* provides the clients that produce human or application useful representations.

The Services: The currently extant services fall into seven categories:

- Measurement Point (MP) service: Creates and/or publishes monitoring information related to active and passive measurements
- Measurement Archive (MA) service: Stores and publishes monitoring information
- Lookup service (LS): Registers all participating services and their capabilities
- Topology service (TS): provides network topology information
- Authentication service (AS): Manages domain-level access to services
- Transformation service (TrS): performs manipulation (aggregation, statistics) on available data sets
- Resource Protector (RP) service: arbitrates the use of limited measurement resources based on the policy of the resource owner.

Use of the System: The Measurement Point (MP) services at the lowest layer create or collect network measurement data. Network operators frequently maintain exclusive management access to their network devices for operational and security reasons. Network operators can use the perfSONAR framework by deploying MP services that query their network devices for state information and push this information into Measurement Archive (MA) services. This provides an important data abstraction function-

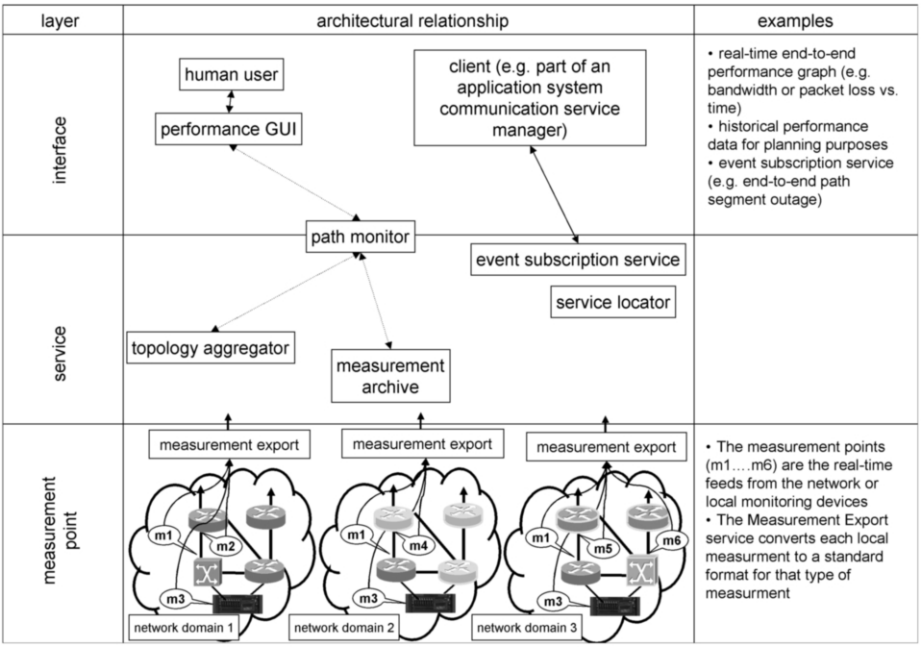


Figure 13. PerfSONAR Architecture.

ality by isolating the method used to obtain the data from the standardized perfSONAR data publication representation. This allows the middle layer of perfSONAR services to process and analyze data from different sources within one domain, or from sources across multiple domains, using a single standardized interface.

This architecture provides a clean separation between the policies regarding how the locally controlled MP accesses the network infrastructure, and the policies governing how internal and external perfSONAR services access the resulting data in the MA services. It has other benefits such as allowing multiple consumers to share the same data thereby reducing the measurement load on the underlying system.

The middle layer of perfSONAR contains a set of cooperating services, including the Measurement Archive (MA), Lookup Service (LS), Topology Service (TS), Transformation service (TrS), and the Authentication service (AS). These services can be used individually, or together to provide uniform access to network measurements across multiple domains.

All services register their presence and capabilities with their local domain's LS. The LS's cooperate to function as a global registry across all domains. This allows the services to find each other within one domain, and it allows applications to find services across multiple domains. The LS allows MP's to locate MA's that can store their results. It allows user applications to locate the MA that contains data of interest.

The TS service supports automated analysis of the network by identifying the underlying structure in the networks and providing information about how multiple network domains are interconnected. This capability will be essential in future networking environments where circuit services will dynamically alter the underlying network infrastructure used by applications in real time.

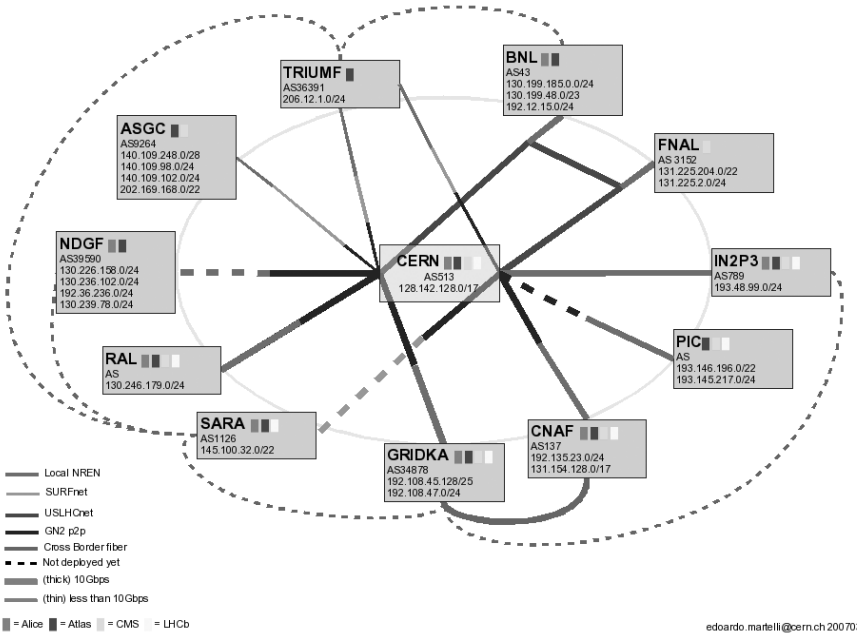


Figure 14. LHC OPN topology showing the physical link elements (see http://lhcopn.web.cern.ch/lhcopn/).

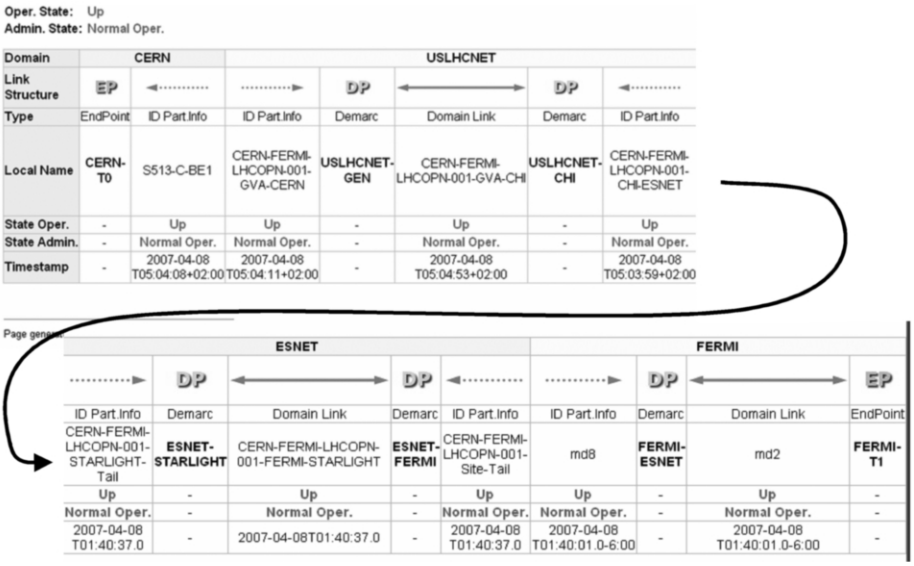
The Measurement Archive (MA) can be configured to accept and store setup requests as well as publication requests. The publication request includes a subscription handle, and the results are sent directly to the client (or indirectly via a TrS). As a client, the MA registers its own presence with an LS, subscribes to an MP, other MA, or TS, and publishes measurement data to subscribers. The MA may send resource availability and authorization requests to the RP.

Multi-Domain Monitoring: The first production deployment of the perfSONAR framework is multi-domain monitoring for the LHC Optical Private Network (LHCOPN or OPN) network (Fig. 14). LHCOPN is the network that transfers data from the LHC Tire-0 facility at CERN to the Tier-1 Data Centers in various countries.

In this case perfSONAR provides a set of conventions for representing network data in a common format, together with the SOA approach that allows the various component services of perfSONAR be used to assemble monitoring applications for different purposes.

perfSONAR MP services are deployed inside each network domain to monitor the links related to each domain's OPN. Some domains are providing real-time status information directly from their MP. Other domains have the MP store the data in a MA, which publishes both current and historical information.

The MP in each domain consists of two components. The domain specific component in the various networks typically interfaces with the operational network monitoring system to obtain the link status data for the portion of the end-to-end path within that particular network. Virtually every network does internal monitoring in a different way that has evolved historically along with the network. The perfSONAR component of each MP takes the resulting data ,generates a standard XML file, and publishes it via



Oper. State: Up
Admin. State: Normal Oper.

Domain	CERN		USLHCNET				
Link Structure	EP	◄·····	·····►	DP	◄——►	DP	◄·····
Type	EndPoint	ID Part.Info	ID Part.Info	Demarc	Domain Link	Demarc	ID Part.Info
Local Name	CERN-T0	S513-C-BE1	CERN-FERMI-LHCOPN-001-GVA-CERN	USLHCNET-GEN	CERN-FERMI-LHCOPN-001-GVA-CHI	USLHCNET-CHI	CERN-FERMI-LHCOPN-001-CHI-ESNET
State Oper.	-	Up	Up	-	Up	-	Up
State Admin.	-	Normal Oper.	Normal Oper.	-	Normal Oper.	-	Normal Oper.
Timestamp	-	2007-04-08 T05:04:08+02:00	2007-04-08 T05:04:11+02:00	-	2007-04-08 T05:04:53+02:00	-	2007-04-08 T05:03:59+02:00

Page gener

ESNET					FERMI			
·····►	DP	◄——►	DP	◄·····	·····►	DP	◄——►	EP
ID Part.Info	Demarc	Domain Link	Demarc	ID Part.Info	ID Part.Info	Demarc	Domain Link	EndPoint
CERN-FERMI-LHCOPN-001-STARLIGHT-Tail	ESNET-STARLIGHT	CERN-FERMI-LHCOPN-001-FERMI-STARLIGHT	ESNET-FERMI	CERN-FERMI-LHCOPN-001-Site-Tail	md8	FERMI-ESNET	md2	FERMI-T1
Up	-	Up	-	Up	Up	-	Up	-
Normal Oper.	-	Normal Oper.	-	Normal Oper.	Normal Oper.	-	Normal Oper.	-
2007-04-08 T01:40:37.0	-	2007-04-08T01:40:37.0	-	2007-04-08 T01:40:37.0	2007-04-08 T01:40:01.0-6:00	-	2007-04-08 T01:40:01.0-6:00	-

Figure 15. E2Emon generated view of the data for one OPN link. Note that the display is split and displayed in two parts for this figure (see http://cnmdev.lrz-muenchen.de/e2e/lhc/G2_E2E_index.html).

the MP service interface, or pushes it to an MA for archiving and publishing. This information is used by an application called E2Emon.[9]

E2Emon uses perfSONAR protocols to retrieve current circuit status every minute or so from MAs and MPs in all domains supporting the circuits.

E2Emon is itself a service that produces Web based, real-time displays of the overall state of the network, and it generates alarms when one of the MP or MA's reports link problems. The web interface for a single link is shown in Fig. 15, and the OPN-wide view is shown in Fig. 16. These tools are being used by the E2ECU (End to End Coordination Unit), which is a function of the GÉANT Network Operations Center that provides the overall management of the OPN circuits.

Another important multi-domain application of perfSONAR is for path performance monitoring. This presents not the just the operational state of the path as in the previous example, but also provide real-time performance such as path utilization and/or packet drop.

Multiple path performance monitoring tools are in development. One example – Traceroute Visualizer[10] – has been deployed at about 10 R&E networks in the US and Europe that have at least some of the required MA services to support the tool. The user input to the tool is a traceroute between elements of a distributed application that defines the path through the IP network. The tool analyzes the path and topology information is retrieved from perfSONAR services; it then queries the MA services in the intervening networks. The MA services returns the requested utilization information, which is passed to a graphing tool. By way of example, the path between Lawrence Berkeley National Laboratory and the Poznan, Poland supercomputer center involves crossing five domain boundaries and is shown in Figs 17 and 18.

[9] An application developed by the German R&E network DFN for monitoring circuits using perfSONAR protocols.

[10] https://performance.es.net/cgi-bin/level0/perfsonar-trace.cgi.

Start page

E2ECU view

All E2E Links
Problem Links

Domain view

CERN
DFN
ESNET
FERMI
GARR
GEANT2
IN2P3
RENATER
SURFNET
SWITCH
USLHCNET

Project view

IGTMD
LHCOPN

Availability Statistics

Current Month
Last Month

All E2E Links

E2E Link ID	State Oper	State Admin	Additional Info
CERN-FERMI-LHCOPN-001	Up	Normal Oper.	
FERMI-IN2P3-IGTMD-002	Down	Normal Oper.	Error: E2E Link is **not** contiguous (End Point missing or gap found) Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
GRIDKA-SARA-LHCOPN-001	Up	Normal Oper.	Error: E2E Link is **not** contiguous (End Point missing or gap found) Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
FERMI-IN2P3-IGTMD-001	Down	Trouble Shooting	Error: E2E Link is **not** contiguous (End Point missing or gap found) Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
GRIDKA-IN2P3-LHCOPN-001	Down	Normal Oper.	Error: E2E Link is **not** contiguous (End Point missing or gap found) Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
CERN-BNL-LHCOPN-001	Up	Normal Oper.	
CERN-SARA-LHCOPN-001	Up	Normal Oper.	
CERN-GRIDKA-LHCOPN-001	Up	Normal Oper.	
CERN-FERMI-LHCOPN-002	Up	Normal Oper.	Error: E2E Link is **not** contiguous (End Point missing or gap found) Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
CERN-SARA-LHCOPN-004	Up	Normal Oper.	Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
CERN-CNAF-LHCOPN-001	Up	Normal Oper.	
CERN-SARA-LHCOPN-002	Up	Normal Oper.	Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
CERN-RAL-LHCOPN-001	Up	Normal Oper.	Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
CERN-BNL-LHCOPN-002	Up	Normal Oper.	Error: E2E Link is **not** contiguous (End Point missing or gap found) Warning: Operational state is known not for all involved links Warning: Administrative state is known not for all involved links
CERN-SARA-LHCOPN-003	Up	Normal Oper.	Warning: Operational state is known not for all involved links

Figure 16. E2Emon generated view of the data for all OPN links showing the operational state and administrative state of each link. (The first entry – CERN-FNAL – is the one line summary of the information presented in the view of Fig. 15.) (See http://cnmdev.lrz-muenchen.de/e2e/lhc/G2_E2E_index.html.)

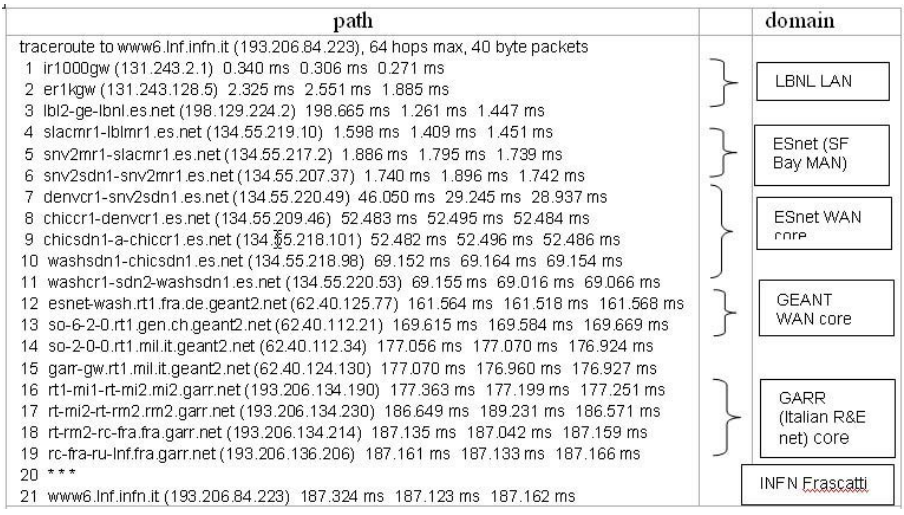


Figure 17. Application view of an end-to-end path.

Status: perfSONAR is being developed through collaboration between some 25 network organizations in US and Europe. The basic framework is complete and the protocols are being documented. New services are being developed and deployed. For more information see [21] and [22].

PerfSONAR is still in its development phases and not yet routinely deployed, though it is gaining ground. Perhaps even more important than the current state of

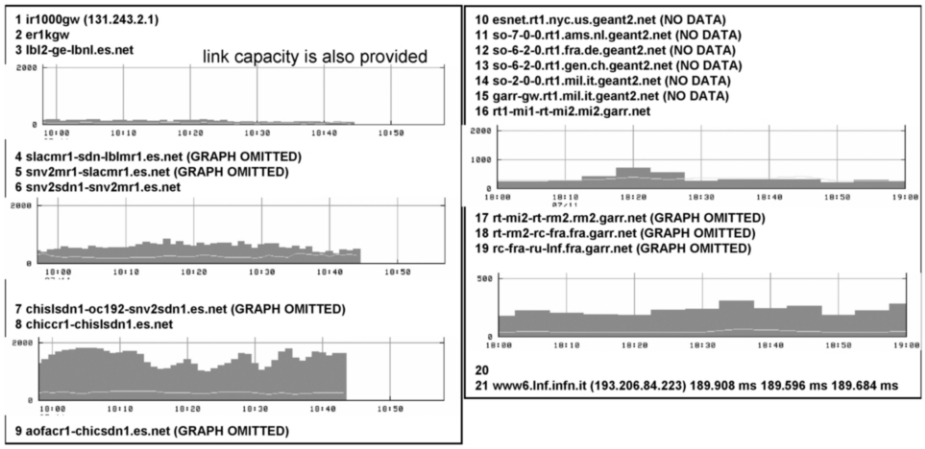


Figure 18. Application path forward (LBNL INFN-Frascati (Italy)) traffic shown as bars on those network device interfaces that have an associated MP services (the first 6 graphs are normalized to 2000 Mb/s, the last to 500 Mb/s).

perfSONAR is the growing recognition within the networking community that the anonymous, best-effort Internet of 10 years ago is no longer adequate to serve the needs of large-scale, data intensive applications such as large scientific instruments and experiments.

Network Outage Footprint Calculator

It is important to solve the problem of determining the impact caused by the failure of a particular network element, and to provide this information to the application.

ESnet has been experimenting with an automated approach to solving this problem. The approach involves two issues: 1) accurately determining the dynamic topology of the network and 2) using the topology to determine current state of the overall network.

Topology Mapping: In order to accurately monitor the network one must accurately model the network. This is accomplished by monitoring each network interface and deriving an accurate IP layer connectivity model of the network on an hourly basis. The daily IP layer connectivity changes that occur through the course of regular operations are captured each night and archived so that retrospective questions about connectivity can be answered.

Outage Footprint Calculator: The Outage Footprint Calculator computes the devices (routers, interfaces) that will be isolated from the network given a list of routers and interfaces out of service.

The current network topology is used to create a list of "vertices" and "edges". A connected graph of the network is derived by inspecting each "edge" in the network topology model and joining the sets of routers at each end of each "edge". During normal conditions when the network is 100% available, the processing of all edges results in a single set of devices representing the fully connected ESnet network as represented by all vertices showing up in a single set of connected vertices.

To compute the effect of removing a set of routers or links, each "edge" connected to the given router(s) or interface argument(s) is removed from the "edge" list prior to

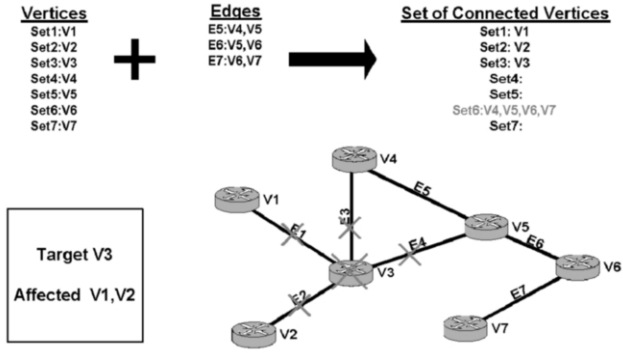


Figure 19. Example of an Outage Footprint Calculation resulting from a single network element failure ("v3").

running the connection algorithm. The resulting affected devices end up in vertex sets that are separate from each other and therefore unreachable. (Each set is a disconnected part of the network.) This is illustrated in Fig. 19. This sort of representation can be combined with a path description in much the same way that is done for the perfSONAR "path monitor" service to provide application-view information about the impact of planed outages in the network.

6. Conclusions

The usage of, and demands on, ESnet (and similar R&E networks) are expanding significantly as large-scale science becomes increasingly dependent on high-performance networking. The motivation for the next generation of ESnet is derived from observations of the current traffic trends and case studies of major science applications. The case studies of the science uses of the network lead to an understanding of the new uses of the network that will be required. These new uses require that the network provide new capabilities and migrate toward network communication as a service-oriented capability. This paper has described ESnet's response to these new directions.

Acknowledgements

The ESnet senior network engineering staff that are responsible for the evolution of ESnet consists of Joseph H. Burrescia, Michael S. Collins, Eli Dart, Jon Dugan, James V. Gagliardi, Chin P. Guok, Yvonne Y. Hines, Joe Metzger, Kevin Oberman and Michael P. O'Connor. This group of people contributed to this paper, with Guok contributing the OSCARS material, Metzger contributing the perfSONAR material, and O'Conner contributing the Outage Footprint Calculator material. Harvey Newman, Caltech, has contributed greatly to ideas about service-oriented network services as needed by the Grid analysis systems of the High Energy Physics community. Thanks are due Brian Tierney of the LBL Distributed Systems Group for applying his expertise in high-performance network based systems and TCP stack tuning during the paper editing process.

ESnet is funded by the US Dept. of Energy, Office of Science, Advanced Scientific Computing Research (ASCR) program. Dan Hitchcock is the ESnet Program Manager and Thomas Ndousse-Fetter is the Program Manager for the network research program that funds the OSCARS project.

ESnet is operated by Lawrence Berkeley National Laboratory, which is operated by the University of California for the US Dept. of Energy under contract DE-AC03-76SF00098.

Notes and References

[1] http://www.energy.gov/, Science and Technology tab.
[2] High Performance Network Planning Workshop, August 2002 http://www.doecollaboratory.org/meetings/hpnpw.
[3] DOE Workshop on Ultra High-Speed Transport Protocols and Network Provisioning for Large-Scale Science Applications, April 2003 http://www.csm.ornl.gov/ghpn/wk2003.
[4] DOE Science Networking Roadmap Meeting, June 2003 http://www.es.net/hypertext/welcome/pr/Roadmap/index.html.
[5] Science Requirements for ESnet Networking, http://www.es.net/hypertext/requirements.html.
[6] LHC Computing Grid Project http://lcg.web.cern.ch/LCG/.
[7] International Committee for Future Accelerators (ICFA), Standing Committee on Inter-Regional Connectivity (SCIC), Professor Harvey Newman newman@hep.caltech.edu, Caltech, Chairperson. ICFA SCIC. Report of the Standing Committee on Inter-Regional Connectivity (SCIC) "Networking for High Energy Physics," February 8, 2007.
[8] http://www.sc.doe.gov/ascr/20040510_hecrtf.pdf (public report).
[9] ASCR Strategic Planning Workshop, July 2003 http://www.fp-mcs.anl.gov/ascr-july03spw.
[10] Planning Workshops-Office of Science Data-Management Strategy, March & May 2004 http://www-user.slac.stanford.edu/rmount/dm-workshop-04/Final-report.pdf.
[11] ESG – Earth System Grid. http://www.earthsystemgrid.org/.
[12] CMS – The Compact Muon Solenoid Technical Proposal. http://cmsdoc.cern.ch/.
[13] The ATLAS Technical Proposal. http://atlasinfo.cern.ch/ATLAS/TP/NEW/HTML/tp9new/tp9.html.
[14] LHC – The Large Hadron Collider Project. http://lhc.web.cern.ch/lhc/general/gen_info.htm.
[15] The BaBar Experiment at SLAC. http://www-public.slac.stanford.edu/babar/.
[16] The D0 Experiment at Fermilab. http://www-d0.fnal.gov/.
[17] The CDF Experiment at Fermilab. http://www-cdf.fnal.gov/.
[18] The Relativistic Heavy Ion Collider at BNL. http://www.bnl.gov/RHIC/.
[19] The Enabling Grids for E-sciencE (EGEE) project is funded by the European Commission and aims to build on recent advances in grid technology and develop a service grid infrastructure which is available to scientists 24 hours-a-day. (http://public.eu-egee.org/).
[20] http://csrc.nist.gov/piv-project/.
[21] "Measuring Circuit Based Networks," Joe Metzger, ESnet. http://www.es.net/pub/esnet-doc/index.html#JM021307.
[22] "ESnet Network Measurements," Joe Metzger, ESnet. http://www.es.net/pub/esnet-doc/index.html#JM021307.

This page intentionally left blank

Chapter 2

Grid Fundamentals

This page intentionally left blank

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

An Infrastructure for the Deployment of e-Science Applications

Wojtek James GOSCINSKI and David ABRAMSON
{wojtek.goscinski, david.abramson}@infotech.monash.edu.au
Faculty of Information Technology, Monash University

Abstract. Recent developments in grid middleware and infrastructure have made it possible for a new generation of scientists, e-Scientists, to envisage and design large-scale computational experiments. However, while scheduling and execution of these experiments has become common, developing, deploying and maintaining application software across a large distributed grid remains a difficult and time consuming task. Without simple application deployment, the potential of grids cannot be realized by grid users. In response, this paper presents the motivation, design, development and demonstration of a framework for grid application deployment. Using this framework, e-Scientists can develop platform-independent parallel applications, characterise and identify suitable computational resources and deploy applications easily.

Keywords. Application deployment, application runtime, grid middleware

Introduction

A computational grid is a High Performance Computing (HPC) infrastructure that enables the development and execution of a new generation of large-scale, multi-domain and multi-institutional e-Science experiments [1]. This new environment, though incredibly promising, poses particular challenges to end-users. One of these challenges is the deployment of software across a computational grid [2, 3].

In order to perform a grid experiment, an e-Scientist first builds an *application testbed* by installing necessary application software across selected resources. While the execution and orchestration of large-scale experiments is becoming more common, users are not provided with a mechanism to install required software across grid resources. Large-scale, dynamic, user-oriented deployment is not supported by current grid middleware. This is a significant problem because the potential scale and highly heterogeneous nature of a grid makes effective manual deployment difficult or even impossible.

In order to most effectively utilize a large-scale grid environment, we believe users require tools and services that allow them to define and realise the largest and most relevant application-testbed. In response, this paper presents the motivation, design, development and demonstration of a framework for grid application deployment. This framework combines three layers of middleware: a user-oriented environment that enables grid-scale application deployment; a platform and data transport independent application-runtime which simplifies deployment over heterogeneous resources; and a deployment tool, which automatically installs an executable and the required library modules.

Using our framework, users can develop platform-independent parallel applications. They deploy those applications using a deployment tool which can characterise resources to form the largest and most relevant testbed. Deployed applications can then be addressed and executed using a deployment reference. As a consequence, users are presented with an overall environment which simplifies the steps required to realize a large-scale grid experiment.

1. Motivation

Consider the following scenario:

> *An e-Scientist with a specialization in biotechnology, and reasonable programming experience, develops a computational model to represent a biological system in the human body. Her experiment is computationally intensive, so strong performance is a high priority. However, considering her specialty in biotechnology rather than computing, she considers simple software development to also be of high importance.*
>
> *She has access to a large set of resources through an inter-university grid but needs to select specific resources which provide the necessary computational, network and storage services. From her point of view, resources are uncharacterised; she knows little about their individual components such as operating systems, architectures or libraries. Her deployment and execution strategy is dynamic and opportunistic; she does not know which resources will be available at the time of execution and she wants to use the most appropriate resources of those available.*
>
> *An analysis of first results shows a small bug in her code. She fixes the bug, recompiles and then redeploys over the entire set of resources. Analysing the results, she continues improving, extending the program and redeploying, to create a consistently better model.*

This simple scenario is relatively difficult to realize with present-day grid infrastructure for a number of reasons:

1. Large-scale, dynamic, user-oriented deployment is not supported in current grid middleware [2, 3]. Users must resort to highly user-intensive command line tools to copy, compile, configure and test software. Deployment is an overlooked feature of grid middleware, but is important for a number of reasons.

   First, the potential scale of application testbeds means it is not feasible to install software on one resource at a time. Second, grid access does not guarantee that a software package can be successfully installed. For example, a grid certificate does not usually allow a user to login to a resource. Furthermore, different resources provide different levels of access, according to the administration policy. For example, while some resources might provide a user with relatively free reign, others might implement strict controls and limitations. Third, users might have access to a grid, but know little about individual resources making it difficult to discover or select appropriate resources.
2. Users are restricted by the vast technical and organizational heterogeneity of the grid. Developing and deploying high performance programs, over a range of heterogeneous resources is a difficult and time consuming process [2-5].

Technical heterogeneity complicates application development and deployment. Standardization of high performance communication libraries, such as MPI, simplifies development for different processors and interconnects. However, programmers still face an assortment of non-MPI porting and installation issues, such as platform specific system calls, recompilation and library problems. Likewise, installation is difficult due to different build and executable files, library dependencies, file systems, build procedures and installation procedures. Furthermore, software usually requires a different installation or build sequence from one resource to another. This is because different resources will support different versions of tools, libraries and compilers. Managing software dependencies in this environment can be a major undertaking.

Further, grids are also organisationally heterogeneous, that is, resources have different administration, operating procedures and access policies. Identical hardware running identical middleware will invariably be configured differently by different system administrators, let alone different institutions. This complicates deployment due to different access, permissions, accounts and installation procedures.

3. Parallel programs are most commonly developed natively; they are written and compiled directly for a specific operating system and instruction set. This approach provides the highest level of performance and control. However, developing and deploying native software is challenging for two reasons:

   First, native code requires porting for different platforms. This is a major undertaking considering the potential complexity of e-Science software and the range of platforms which might constitute a grid. Furthermore, native code needs to be recompiled for each platform, which itself is a significant effort. This limits users when making deployment and execution decisions. Second, native binaries alone provide no mechanisms for defining library dependencies. Users need to know an application's dependencies and manually install them.

   This is in contrast to the current trend in software engineering, which has moved toward an application-runtime architecture, such as that provided by Java [6] or the Common Language Infrastructure [7] and its commercial implementation, Microsoft .Net [8].

   Typically, native code is faster in comparison to code managed by a runtime environment. On the other hand, runtime environments provide a range of services which have proved successful in fostering software development in the general areas of business and desktop computing. These include: compile-once-run-anywhere development, a common type system, platform-independent self-describing assemblies and a guaranteed set of libraries.

   Most significantly, runtime environments provide an execution guarantee. In essence, a runtime specification defines a contract between user code and the runtime. As a result, code is guaranteed to execute on a correctly implemented runtime, so long as both parties adhere to the contract. This allows developers to focus on functionality, rather than porting to various platforms. It is particularly significant for an e-Scientist installing software over a range of heterogeneous resources.

   Unfortunately, existing runtimes alone do not provide the necessary functionality for building HPC applications, particularly message passing and shared memory. We believe this is a significant factor which has prevented e-Scientists from using runtime environments for parallel processing.

2. Infrastructure for the Deployment of e-Science Applications

In response to the deficiencies introduced in the preceding section we present the design and implementation of an Infrastructure for the Deployment of e-Science Applications (IDEA). This framework consists of three middleware layers (Figure 1):

1. Distributed Ant (DistAnt) [2, 3], a user-oriented service-level infrastructure for grid-scale software deployment over heterogeneous uncharacterised resources. It allows developers to define and realize a testbed and consists of two major components: the DistAnt Service which provides a common deployment service (Section 2.3); and the DistAnt Workflow which is a tool for describing and enacting deployment steps (Section 2.4).
2. Motor [4], an application-runtime which provides a common environment for software development, deployment and execution. It supports a runtime-internal MPI communication library for parallel processing.
3. The Automatic Deployment Tool (ADT), a tool that automatically deploys a Motor executable and its library dependencies.

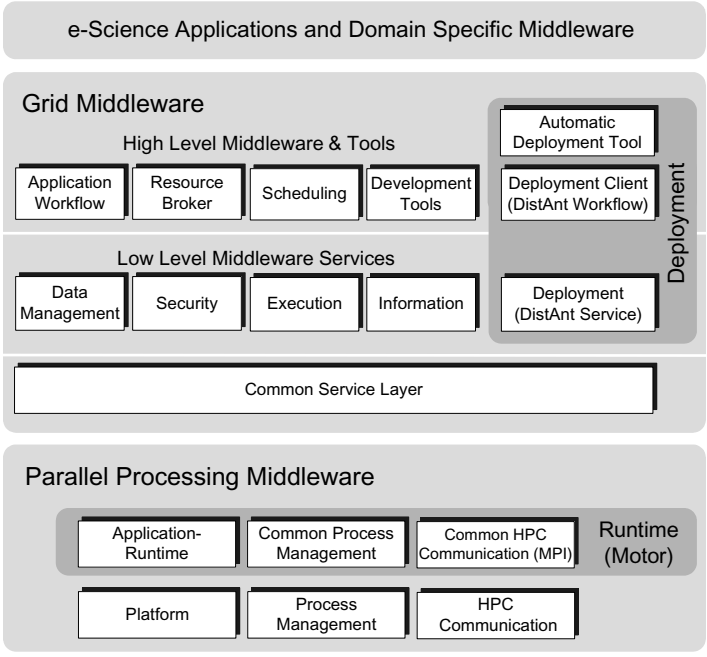


Figure 1. The multilayer IDEA framework, showing its relationship within existing grid and HPC middleware layers. Components defined by the IDEA framework are highlighted in dark grey.

The following sections describe important features and advantages of this architecture and its components. Section 3 describes the use of the whole IDEA framework in developing and deploying parallel applications.

2.1. User-Oriented Application Deployment

Application deployment should be simple; otherwise, the potential of grids will not be realized. The user-oriented nature of the grid requires that users have the ability and authority to deploy software on resources to which they have received access, but do not necessarily control. To support simple application deployment, our architecture is *user-oriented*, which means the motivation and responsibility for a successful deployment comes from the user, with support from middleware. This model is appropriate because the grid is decentralised and multi-organisational. Relying on system administrators to install software is difficult because there is no single point of control.

Our user-oriented system, DistAnt, supports automatically characterising a grid, managing heterogeneity and deploying software, thus allowing e-Scientists to form testbeds easily. The way this is achieved is described in later sections.

2.2. Dynamic Application Testbeds

A deployment system assists users in creating large and dynamic testbeds, which is important because it provides the scheduler with the largest possible choice of targets. In addition, testbeds need to be dynamic because grids are physically dynamic. For example: new resources might become available or existing resources might be upgraded, changed, replaced or retired. In the majority of cases, users have no control over the architecture or characteristics of the grid and its resources.

Currently, testbeds are relatively static and limited in size. This is because deployment, the major task to forming a testbed, is manual and cumbersome; it is difficult to create, augment or adapt a testbed. For example, the US based TeraGrid [9] consists of a static and relative small number of high end resources. Moreover, in order to try and simplify the deployment problem, the environment on these machines is fixed and heavily controlled.

Consider the example of a new cluster being introduced into a grid. For the scheduler to take advantage of this resource, the users must discover the new cluster and integrate it into their testbed, by manually installing necessary software.

A deployment system improves the outcomes of scheduling and execution because it can support the creation of larger, more relevant and more dynamic testbeds. For example, consider the following two scenarios:

- *The execution of a parameter sweep.* To minimize execution time the application needs to be deployed over as many resources as possible. An automatic deployment system could be used to install the application over a larger set of resources than would be otherwise possible manually.
- *The execution of a scientific application which requires a distinctive hardware specification.* Resources must be identified which match the specification. However, users should not be expected to know details about every resource in a large-scale grid and manually characterising resources is time consuming. An automatic deployment system could be used to characterise and select the most appropriate resources making the testbed more relevant.

In both cases, the testbed is dynamic because deployment can be revisited. Thus, a user can automatically adjust the testbed to changes in the physical grid environment.

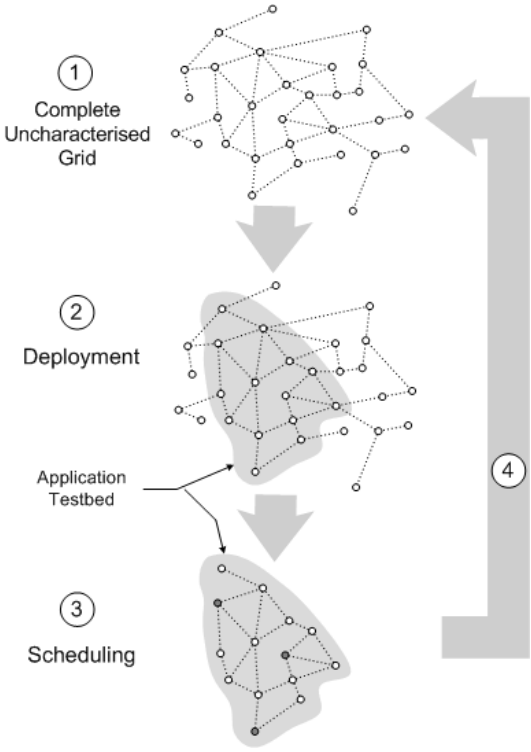


Figure 2. The interaction between automatic deployment for building dynamic testbeds and scheduling.

Our system, DistAnt, enables the following deployment, scheduling and redeployment steps (Figure 2):

1. The starting point is a completely uncharacterized grid. Therefore, the grid is large, but not necessarily relevant to the experiment because resources are unknown and do not support the required software.
2. The DistAnt system characterizes resources. It defines the testbed by selecting resources based on their physical attributes and characteristics required by the user.

   The DistAnt system realizes the testbed by deploying the necessary software over the selected resources. The testbed is relevant to the requirements of the experiment, based on the current physical state of the grid.
3. Scheduling is performed based on the runtime performance of the realized testbed, according to the appropriate scheduling strategy. Earlier steps allow the user to select resources so that scheduling is most effective. For example, the testbed for a parameter sweep should be as large as possible.
4. Revisiting deployment allows the user to update the testbed to reflect changes in the physical grid environment, software requirements, experiment requirements, or simply to upgrade software. By re-characterising the grid environment a new testbed can be formed which reflects the updated grid.

2.3. An Application Deployment Service

Much of the effort invested in developing grid middleware is devoted to defining common interfaces for resource services. The importance of this effort is demonstrated in the success of the Globus GRAM [10], which provides a common interface for execution. Similarly, deployment also requires a common service interface. Currently, to install an application, a user must log in and use supported services such as FTP, gridftp, telnet and SSH. Each resource might support different services and enforce different authentication and file system access mechanisms.

The DistAnt Service provides a common, secure access point and a standard way of performing deployment steps, such as: transporting and unpacking files, compiling and linking binaries, and enacting necessary configuration steps. Like the Globus GRAM, this service supports a user-oriented view of deployment, allowing users with resource access to install software.

The DistAnt Service also provides *deployment spaces*, which are managed local directories where users can install software. Deployment spaces can be referenced using a *deployment reference* which is a user defined handle. The combination of deployment spaces and deployment references allows users to reference an application across a set of resources regardless of the actual underlying file system location or file system structure.

The DistAnt Service can be invoked by a range of tools or clients that wish to deploy software on the hosting resource. Our design of such as tool is the DistAnt Workflow, which is described in the following section.

2.4. An Application Deployment Workflow

An application is deployed by performing a series of steps, such as file transfer, compilation, and configuration. We refer to this sequence of steps as the *deployment workflow*. Consider the following two examples:

- *A managed application with one executable binary.* In this case, the workflow needs to identify resources which support the necessary runtime and then select the most relevant resources based on the particular deployment strategy. Realizing the testbed requires copying the binary to selected resources.
- *A complex native application which is dependant on a number of libraries and can be installed from source over a range of Linux distributions.* In this case, the deployment workflow needs to identify Linux resources which support the necessary libraries and then select the most relevant resources based on the particular deployment strategy. Realizing the testbed could require a complicated series of steps on each testbed resource, such as: moving and unpacking files, performing platform specific compilation and linking, installing missing dependencies and performing a resource-specific configuration.

Performing these steps manually over a large testbed is no longer realistic. Therefore, DistAnt allows the user to describe and enact a deployment workflow. The DistAnt Workflow provides the user with a flexible mechanism for describing and enacting deployment steps, including:

- The steps for *defining* the testbed. This includes characterising and selecting grid resources. Users can write a workflow which finds those grid resources which match requirements and group them to form a testbed; *and*

- The steps for *realizing* the testbed by installing necessary software over the set of selected resources. The DistAnt Workflow provides constructs for unpacking, compiling, linking and configuring software at the remote resource. In addition, the workflow allows the user to manage resource heterogeneity. For example, it is straightforward to write a workflow which performs a different series of configuration steps, depending on the target platform.

The workflow description is built as an extension to the Ant build file system [11] and is both *procedural* and *declarative*. Procedural scripting allows the user to define deployment routines, such as the steps required to realize an installation. The declarative construct allows the user to define dependencies between routines. Each routine defines the steps for performing a particular action on the client resource *or* over a set of remote resources. Examples of specific routines might include: querying resource attributes to characterise a set of resources; forming grids into logical sets; transferring files; or enacting remote actions to perform configurations.

To support the user performing software configurations on remote resources the DistAnt Workflow provides remote routines that describe tasks such as unpacking, compiling, linking, configuring software or executing a script at the specific resource. The resource-side functionality for enacting a remote routine is provided by the DistAnt Service.

In addition, the DistAnt Workflow provides a way of managing large scale and heterogeneous grids, as discussed in the following section.

The DistAnt Workflow is a high-level middleware tool (Figure 1); it invokes a range of grid middleware services. For example, deployment functionality is provided by the DistAnt Service, file transfer is provided by a file transfer service such as GridFTP, RFT and SCP and resource information is provided by an underlying information service, such as the GT4 Index Service.

A comprehensive description of the DistAnt Workflow tool, the operations and tasks it supports, and the way a user can write a DistAnt workflow is provided in [2, 3].

2.5. Resource Heterogeneity

In order to successfully deploy software, a user needs to consider a range of resource attributes and characteristics, such as:

- The execution environment, including the operating system, architecture, configuration, compiler, binary format and data transport. Native software is often written for a specific execution environment, while code which is developed for a range of platforms must be recompiled for each platform. This is time consuming and error prone across a large testbed.
- Underlying physical resource attributes, including hardware metrics such as CPU speed, real memory size, hard disk space or distinct and specialized hardware. These attributes and metrics are important considerations when deploying software. For example, an application might require a minimum amount of real memory to execute correctly.
- HPC attributes such as cluster architecture or number of compute nodes. For example, a parallel program might need to be configured for the number of compute nodes before compilation.

- Attributes specific to the overarching experiment or workflow. For example, the file system presence of a particular data set might be a requirement of execution and therefore a prerequisite to deployment.
- Grid, organisation or resource-specific management policies. Resources will be managed differently by different organisations. For example, maximum user account disk space limits the maximum installation size.

In a typical multi-organisational grid environment, these attributes are beyond the control of the user and typical ussers will know little about individual resources, making it difficult to discover and select appropriate resources to form a relevant testbed. Furthermore, once a testbed is defined, it is likely that different resources will require a specific installation procedure. For example, software deployed on a Linux resource will require a Linux-specific compilation.

Our framework supports two approaches to developing and deploying software over heterogenous grids:

1. Native software development and deployment, with support from the DistAnt system to manage heterogeneity. This means developing, compiling and configuring software for the specific platform supported by each particular resource.
2. Managed software development and deployment. This means software is developed for Motor, our execution environment, which vastly simplifies deployment.

While the IDEA framework does include a runtime for parallel program development, we do not want to abandon users who develop native software. Developers will continue writing native parallel codes because this approach offers strong performance and highest level of control. As a result, we believe it is essential that a deployment system is not limited to managed applications, but also benefit new and legacy native applications.

It is important to note that the majority of the middleware for deploying native software remains essential for deploying managed software. Characterizing resources, selecting resources, transporting files, performing installation and configuring must be performed, regardless of whether Motor is used.

Section 2.5.1 discusses the mechanisms supported by DistAnt to manage resource heterogeneity, while Section 2.5.2 presents the alternative approach, virtualization.

2.5.1. Managing Heterogeneity using DistAnt

Using the DistAnt Workflow resource attributes are automatically discoverable and can be managed. To support this, the DistAnt Workflow defines two concepts:

- Resource queries; and
- A subgrid, which groups resources together in logical sets.

Resource Queries

The DistAnt Workflow characterises resources using a query mechanism. This is intended to be performed over a whole set of resources. For example, to discover the CPU speed of a set of resources, a user would query for CPU speed. This query would be executed for each resource in the set and the specific value would be assigned as a property to a workflow representation of each resource.

Queries are executed against a particular grid information service or other interface which can be queried for information. Currently, DistAnt supports WSRF queries. For

example, it is easy to query the GT4 Index service for resource information.

Subgrids

An effective approach for managing a large heterogeneous grid is to sub-divide it into manageable sets of homogeneous resources, which can then be treated individually. This allows the user to write routines for specific resource attributes. For example, resources which share the same platform description are likely to require a similar sequence of build steps.

The DistAnt Workflow allows the user to characterise, group and select platforms without specifically addressing individual resources. To achieve this, DistAnt implements a construct called a *subgrid*, which makes it possible to group like resources together. A subgrid is a set of zero or more resources which share specific properties. Examples of subgrids might include: the set of resources which require a particular library upgrade; or a set of resources which support the same operating system.

Defining a subgrid allows the user to write specific routines and functionality for that subgrid. For example, a user might choose to group Linux-X86 resources together in a common subgrid to perform a common Linux-X86 configuration. Significantly, a user can write a deployment workflow without addressing individual resources. Rather, the workflow can address sets of resources based on their properties. This is important because individual resources might come and go, but types of resources are likely to remain stable.

In our experience, e-Science programs are commonly installed from source code. However, a complete build from source on each individual resource is typically unnecessary. Therefore, the subgrid construct has been designed so that it is easy to build from source on one resource and redistribute binary to remaining resources. To achieve this, a subgrid can be assigned a representative resource, which means source can be compiled once on the chosen resource for each platform and then redistributed to remaining resources of that platform. Selecting a representative resource can be based on resource properties. For example, CPU speed or resource load could be used to choose the fastest resource to compile an application.

Figure 3 illustrates how the DistAnt Workflow can be used to characterise a grid and then use subgrids to build source and redistribute the platform-specific binary. The figure illustrates the following three steps:

1. The starting point is an uncharacterized grid.
2. Resources are characterised and grouped into platform-specific subgrids.
3. A representative resource is chosen from each subgrid. This resource performs the compilation and the completed deployment is then archived and redistributed to other resources of that subgrid.

Deploying native software, from source code, across a heterogeneous grid using DistAnt has been demonstrated in a major deployment case study [3]. Using DistAnt, the GAMESS quantum chemistry package and a FORTRAN compiler were installed from source across a testbed of 4 different platform types. A DistAnt workflow was written which characterised the available resources, deployed GAMESS source to the fastest resource of each platform, preprocessed and compiled the source and then redistributed the binary to the whole testbed. If no resources of a particular platform supported a FORTRAN compiler then the gcc-g77 FORTRAN compiler was automatically installed. The deployment was performed with no changes to GAMESS code, very minimal changes to the existing GAMESS build file, and no changes to gcc-g77 code

or build file. The deployment workflow was around 300 lines long and a complete deployment over 8 selected resources took 92 minutes.

This case study demonstrated that a DistAnt workflow could be used to build and deploy software quickly over a uncharacterized grid. In addition, once a deployment workflow was written, it was easily executed, repeatable and non-user-intensive.

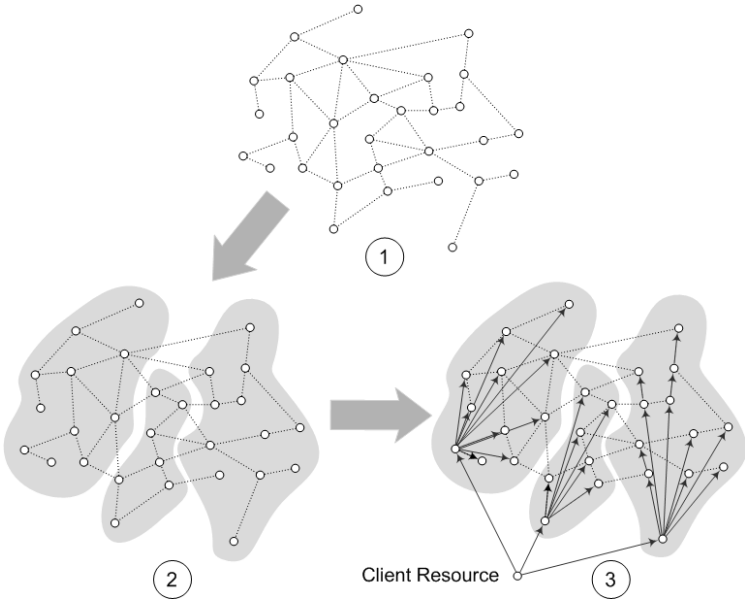


Figure 3. The logical sequence of steps to build source and redistribute binaries using subgrids.

2.5.2. Managing Heterogeneity through Virtualization

Deployment can also be simplified if resources support a common execution environment. In this way, software can be written and built for the virtual environment, rather than a range of specific native platforms, libraries, compilers and environments. Deployment decisions can be made without considering platform and system heterogeneity.

2.5.3. The Motor Application-Runtime

Motor [4] is an application-runtime for HPC, with a runtime-internal communication library. It is an environment based on the Common Language Infrastructure (CLI)[1] [7] and supports two sets of HPC message passing operations:

1. A standard set of MPI-like operations. These provide an efficient message passing mechanism, integrated within the runtime managed memory environment.

[1] The CLI is a Microsoft sponsored specification which defines a language-independent application-runtime environment. It is better known by its commercial implementation, .Net. A range of compilers have been developed which target the CLI environment, including: C#, FORTRAN, C++, Perl, LISP, Java, Visual Basic and C. The SSCLI is the 'shared source' reference implementation of the CLI.

2. A set of higher-level operations for structured data transport. These provide a convenient mechanism to *automatically* transport objects, arrays and object trees.

The Motor implementation is a synthesis of three parts: the Microsoft Shared Source CLI (SSCLI)[1] [12], which provides an existing runtime; MPICH2 which provides a portable MPI implementation; and an additional set of components which implement the higher level message passing operations, and interface to the integrated MPICH2 library. A detailed description of the Motor implementation and its performance is provided in [4]. This paper provides an overview of the Motor architecture because it provides an environment with a guaranteed HPC communication library and is an important part of our deployment framework.

The Motor Runtime-Internal Architecture

A number of environments provide a Java or .Net parallel communication mechanism. These projects can be grouped into two categories:

1. *Pure managed* implementations which execute entirely over the virtual machine. These include JMPI [13] and jmpi [14]. Because they are written entirely in Java they are portable, but are also inefficient.
2. *Runtime-external* implementations which provide a managed interface to an underlying native communication facility. This includes systems such as mpiJava [15], JavaMPI [16], MPIJ [17] and the Indiana .Net bindings [18]. The runtime-external message passing systems perform significantly better than pure managed systems because the high performance communication mechanism has direct native access to transport services.

In both of these classifications the runtime environment is unaware of the HPC communication library. In contrast, the Motor architecture is *runtime-internal*, whereby the runtime environment provides a shared memory or message passing abstraction.

Environments such as Java and the CLI provide web service, user interface, security and networking libraries as *standard* because these are essential for building business and desktop software. We believe that a *high performance* runtime environment design should follow the same philosophy and include essential libraries as standard. Therefore, Motor extends an existing runtime environment with high performance communication, thereby creating a *complete* runtime environment for HPC. The Motor runtime-internal architecture is illustrated in Figure 4, in contrast to the runtime-external architecture.

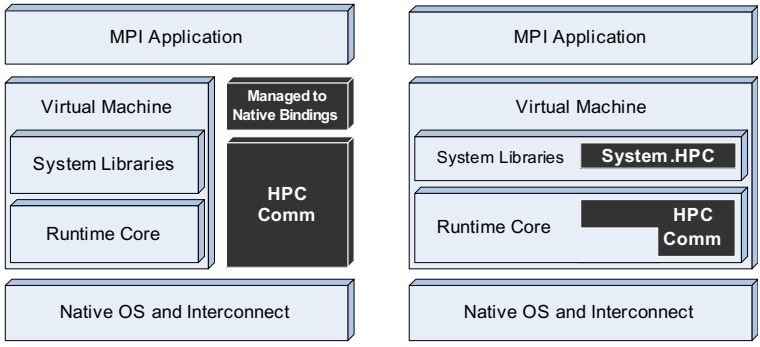


Figure 4. The runtime-external architecture (left), in contrast to the Motor runtime-internal architecture (right).

A *complete* runtime environment is important both from the developer's perspective and is also important for deployment; it is a guarantee that the runtime will support a well defined high performance communication abstraction. In contrast, managed-wrapper MPI implementations must be rebuilt to interface to a resource-specific native MPI implementation. The message passing library is an afterthought, rather than a service guaranteed by the runtime library.

In addition, there are a number of other major reasons why the Motor runtime-internal design is preferable:

1. Runtime-external implementations lack portability – the interface must be rebuilt and sometimes edited for each different underlying native implementation. Therefore, this architecture conflicts with the compile-once-run-anywhere ethos of managed software development. In a runtime-internal implementation, much of the implementation can be defined in higher layers so that only low-level communication primitives and process-management-specific code need to be re-implemented.
2. Implementing the communication abstraction internally results in better performance than the alternative runtime-external architecture. In micro-benchmarks the Motor MPI library has performed noticeably better than runtime-external MPI implementations [4]. For example, in a ping-pong micro-benchmark Motor performs better than runtime-external bindings hosted by the SSCLI using MPICH2; 16% at a peak; 8% on average over all buffer sizes; and 3% on average over buffer sizes greater than 65,536 bytes. These are positive results considering Motor is a synthesis of these same major components: SSCLI and MPICH2 [4]. In addition we have tested Motor using a real application, Shallow Water which is an implementation of the shallow water model [19]. We implemented a C version of Shallow Water which is executed using the Motor environment. In these results [20], Motor performs favorably against the SSCLI, using runtime-external bindings. On average Motor performs 21.9% faster than SSCLI and only 4.5% slower than the native implementation.
3. A runtime-internal implementation has full access to internal runtime services, such as the runtime thread model, memory management, and the object and class model. The advantage of this is that the environment can support higher levels of functionality, such as the Motor structured data transport operations. The result is also performance. In ping-pong micro-benchmarks the Motor structured data transport operations performed significantly better than similar runtime-external MPI implementations using runtime-provided serialization [4].

2.6. The Application Development Lifecycle

Deployment is an integral part of the overall software development, testing and execution lifecycle. Because DistAnt is an extension of Ant, it allows a user to write a deployment workflow as part of the build file and test software on a testbed, during and after the development phase.

It is important that deployed software is easy to execute. Therefore, the DistAnt system has been integrated with a high-level resource broker and scheduling environment, Nimrod/G [21]. This allows users to write regular Nimrod/G execution descriptions and reference a DistAnt deployment using the deployment reference [3].

2.7. Dependency Management

Managing native software dependencies is a challenging task over a large-scale heterogeneous grid. For example, different resources support different libraries versions or compiler implementations. Because the DistAnt Workflow is built using the Ant build file system, it allows the developer to manually define software dependencies in the same way as is provided by Ant.

In addition our framework provides the Automatic deployment Tool (ADT), which automatically generates and enacts a workflow for deploying a Motor executable and its libraries over a testbed. It performs this deployment in two steps. First, it queries Motor binaries recursively requesting dependency information. Because Motor is based on the CLI, it supports self-describing binaries which contain information about library dependencies. Second, it forms and enacts a DistAnt Workflow based on the computed dependency tree, thus deploying the executable and all its libraries over the grid testbed.

However, not every resource will require each binary. Some resources might already support one or more application components, perhaps due to an earlier deployment. Therefore, the ADT generates a workflow which queries resources for the presence of necessary binaries. It groups resources into subgrids which require particular binaries and deploys those binaries only to the required subgrids.

The result of the ADT is that users do not need to manually define library dependencies.

3. Middleware Supported Deployment

Our framework and its components support the following software development, deployment and execution steps (Figure 5):

1. *Software development*. Users have the option of writing managed code for the Motor application-runtime rather than porting native code for a range of platforms or environments.
2. *Software compilation.* Motor application-runtime compiled binaries are:
   - Process-independent, which enables compile-once-run-anywhere deployment. This means that users can compile code once for all resources, rather than once for each platform; and
   - Self-describing, which allows management of library dependencies using the ADT.

   Alternatively, DistAnt supports users to compile native software across a range of platforms by compiling on one of each platform and redeploying the resulting binary.
3. *Define the application testbed.* The user writes a DistAnt workflow which characterises grid resources. Similar resources can be grouped together based on these same attributes using the subgrid construct. Therefore, the user can easily define a testbed by selecting resources which match resource attributes to software and experiment requirements.
4. *Realize the application testbed*. The user realizes the testbed by writing a deployment workflow which deploys the given application. Alternatively, a deployment workflow can be automatically generated based on a runtime executable and its library dependencies.

The deployment service provides facilities to perform installation and configuration procedures on each remote resource. Different remote actions can be performed at different resources by defining subgrids and specific deployment routines.

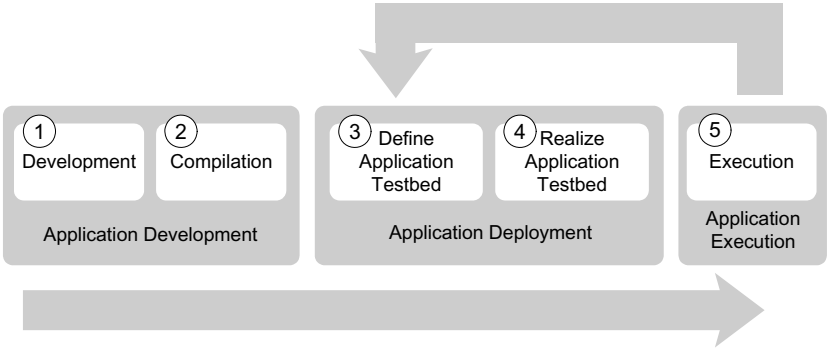


Figure 5. Software development, deployment and instantiation steps using the IDEA framework.

5. *Execution*. A resource management system or higher level grid service, such as Nimrod/G, is used to schedule execution over the testbed. DistAnt deployment references provide an execution handle to the deployment across the entire testbed. The HPC runtime ensures execution.

Importantly, these steps are easily revisited, allowing the user to redefine the testbed and adapt to changes in the physical grid architecture, software requirements or experiment requirements.

4. Related Work

A number of projects have focused on providing a suitable runtime environment for large-scale e-Science applications [21-27]. For example, the Globus project defines middleware interfaces that support application invocation in a secure, distributed, environment [25]. However, none of these environments specifically address deployment over a large-scale distributed and heterogeneous grid.

System configuration and management provide the functionality to transform a blank machine into a functional system [28-30] including many proprietary systems [30]. Similarly, component and object-oriented management systems, allow the user to define components and their relationships in order to automatically generate the described application system [31, 32].

However, these configuration management systems and component management systems are *super-user-oriented* systems; they define a *centralized* mechanism providing administration and configuration over systems within institutional control. This is an evolution of cluster management where deployment is largely a non-issue; users typically only need to deploy software once, on a shared file system, so that it can be executed across the entire cluster. In contrast, deploying software over a loosely coupled set of grid resources requires an individual deployment on each resource. Furthermore, a multi-institutional grid has no one overarching super-user to control software issues such as dependencies, version issues or clashing libraries.

Some grid service hosting and deployment systems, provide a facility to 'hot' deploy grid services [33-39]. None of these systems support deployment of non-hosted software, such as native parallel applications. Nor do they deal with other user-oriented issues such as scale and heterogeneity, which we have identified as significant issues.

In addition, platform-level virtualization systems, provide an infrastructure to host application using a platform-level virtual machine such as VMWare or Xen [40-46]. This approach requires a very large effort to deploy even the simplest of software. For example, to deploy relatively simple code an entire operating system image would need to be generated, configured or transported to the remote resource. Furthermore, platform-level deployment does not address specifics of the operating system, virtual or otherwise.

5. Conclusion

The outcome of our framework is that users are provided with an overall environment which supports the creation of application testbeds.

DistAnt is user-oriented allowing a regular user to deploy a native or managed application over an uncharacterized and heterogeneous set of resources. The DistAnt Service is significant because it defines a common access point and interface for deployment, while the DistAnt Workflow provides a facility to describe and enact the steps to define and realize a testbed.

In addition, Motor provides a common environment for high performance applications. Thus, users can develop and compile application software for the Motor environment rather than a range of platforms, simplifying application development and deployment. The integration of a message passing library guarantees that the runtime will support a well known parallel communication mechanism. Finally, the ADT is significant because it automatically deploys a Motor executable and its libraries.

References

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, 1st edition ed: Morgan Kaufmann Publishers, 1998.
[2] W. Goscinski and D. Abramson, "Distributed Ant: A System to Support Application Deployment in the Grid" *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pp. 436 - 443, 2004.
[3] W. Goscinski and D. Abramson, "Application Deployment over Heterogeneous Grids using Distributed Ant" *First International Conference on e-Science and Grid Technologies (e-Science 2005)*, pp. 361 - 368, 2005.
[4] W. Goscinski and D. Abramson, "Motor: A Virtual Machine for High Performance Computing" *The Fifteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC2006)*, 2006.
[5] D. Abramson, "Applications Development for the Computational Grid" *Lecture Notes in Computer Science*, vol. 3841, 2006.
[6] "Java", 2006, http://java.sun.com.
[7] "Standard ECMA-335: Common Language Infrastructure, 3rd edition (June 2005)", 2005, http://www.ecma-international.org/publications/standards/Ecma-335.htm.
[8] "Microsoft.NET", 2006, http://www.microsoft.com/net/.
[9] "TeraGrid", 2006, http://www.teragrid.org/.
[10] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems" *Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[11] "Apache Ant", 2006, http://ant.apache.org/.
[12] "Shared Source Common Language Infrastructure 1.0 Release", 2003, http://msdn.microsoft.com/net/sscli/.
[13] S. Morin, I. Koren, and C. Krishna, "JMPI: Implementing the message passing standard in Java" *International Parallel and Distributed Processing Symposium: IPDPS 2002 Workshops.*, 2002.
[14] K. Dincer, "Ubiquitous message passing interface implementation in Java: jmpi" *Proceedings of the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, 1999.
[15] M. Baker, B. Carpenter, G. Fox, S. H. Ko, and S. Lim, "mpiJava: An Object-Oriented Java interface to MPI" *International Workshop on Java for Parallel and Distributed Computing*, 1999.
[16] S. Mintchev, "Writing programs in JavaMPI" *Technical Report MAN-CSPE-02*, 1997.
[17] G. Judd, M. Clement, Q. Snell, and V. Getov, "Design issues for efficient implementation of MPI in Java" *Proceedings of the ACM 1999 Java Grande Conference*, 1999.
[18] J. Willcock, A. Lumsdaine, and A. Robison, "Using MPI with C# and the Common Language Infrastructure" *Concurrency and Computation: Practice and Experience*, 2005.
[19] C. Vreugdenhil, *Numerical Methods for Shallow-Water Flow*: Kluwer Academic Publishers, 1994.
[20] W. Goscinski, "IDEA: An Infrastructure for the Deployment of e-Science Applications" *Monash University PhD Thesis*, 2007.
[21] D. Abramson, J. Giddy, and L. Kotler, "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?" *International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 520 - 528, 2000.
[22] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids" *Proceedings. 10th IEEE International Symposium on High Performance Distributed Computing*, 2001.
[23] F. Berman, A. Chien, K. Cooper, J. Dongarra, Ian Foster, D. Gannon, L. Johnsson, K. Kennedy, Carl Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski, "The GrADS Project: Software Support for High-Level Grid Application Development" *International Journal of High Performance Computing Applications*, vol. 15, 2001.
[24] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, "The WS-Resource Framework", 2004, http://www.globus.org/wsrf/specs/ws-wsrf.pdf.
[25] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit" *The International Journal of Supercomputer Applications and High Performance Computing*, 1996.
[26] I. Taylor, I. Wang, M. Shields, and S. Majithia, "Distributed computing with Triana on the Grid" *In Concurrency and Computation:Practice and Experience*, vol. 17(1-18), 2005.
[27] R. Stevens, A. Robinson, and C. A. Goble, "myGrid: Personalised Bioinformatics on the Information Grid" *11th International Conference on Intelligent Systems in Molecular Biology*, 2003.
[28] G. Bruno, P. Papadopoulos, and M. Katz, "Npaci rocks: Tools and techniques for easily deploying manageable linux clusters" *3rd IEEE International Conference on Cluster Computing (CLUSTER'01)*, 2001.
[29] P. Anderson, G. Beckett, K. Kavoussanakis, G. Mecheneau, J. Paterson, and P. Toft, "Large-Scale System Configuration with LCFG and SmartFrog", 2003, http://www.gridweaver.org/WP3/report3_1.pdf.
[30] P. Anderson and A. Scobie, "LCFG: The Next Generation" *UKUUG Winter Conference*, 2002, http://www.lcfg.org/doc/ukuug2002.pdf.
[31] Patrick Goldsack, Julio Guijarro, Antonio Lain, Guillaume Mecheneau, Paul Murray, and P. Toft, "SmartFrog: Configuration and Automatic Ignition of Distributed Applications", 2003.
[32] P. Anderson, G. Beckett, K. Kavoussanakis, G. Mecheneau, and P. Toft, "Technologies for Large-Scale Configuration Management", 2002, http://www.gridweaver.org.
[33] D. Kurzyniec, T. Wrzosek, D. Drzewiecki, and V. Sunderam, "Towards self-organizing distributed computing frameworks: The H2O approach" *Parallel Processing Letters*, vol. 13, 2003.
[34] M. Migliardi, V. Sunderam, A. Geist, and J. Dongarra, "Dynamic reconfiguration and virtual machine management in the Harness metacomputing system" *Lecture Notes in Computer Science*, vol. 1505, 1998.
[35] L. Qi, H. Jin, I. Foster, and J. Gawor., "HAND: Highly Available Dynamic Deployment Infrastructure for Globus Toolkit 4", 2006, http://www.globus.org/alliance/publications/papers/HAND-Submitted.pdf.
[36] J. Weissman, S. Kim, and D. England, "A Framework for Dynamic Service Adaptation in the Grid: Next Generation Software Program Progress Report" *19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, 2005.
[37] J. Weissman, S. Kim, and D. England, "Supporting the Dynamic Grid Service Lifecycle" *5th International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, 2005.

[38] G. Wasson, N. Beekwilder, M. Morgan, and M. Humphrey, "OGSI.NET: OGSI-compliance on the.NET framework" *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2004)*, 2004.
[39] E.-K. Byun, J.-W. Jang, W. Jung, and J.-S. Kim, "A Dynamic Grid Services Deployment Mechanism for On-Demand Resource Provisioning" *5th International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, 2005.
[40] I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo, "VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing" *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, 2004.
[41] K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron, "Virtual Workspaces in the Grid" *11th International Euro-Par Conference*, 2005.
[42] X. Zhang, K. Keahey, I. Foster, and T. Freeman, "Virtual Cluster Workspaces for Grid Applications " *ANL technical report, ANL/MCS-P1246-0405*, 2005.
[43] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang, "Virtual Clusters for Grid Communities" *IEEE International Symposium on Cluster Computing and the Grid*, 2006.
[44] R. Figueiredo, P. Dinda, and J. Fortes, "A Case For Grid Computing On Virtual Machines" *Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003.
[45] S. Adabala, V. Chadha, P. Chawla, R. J. O. Figueiredo, J. A. B. Fortes, I. Krsul, A. M. Matsunaga, M. O. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu, "From virtualized resources to virtual computing grids: the In-VIGO system" *Future Generation Computer Systems*, 2005.
[46] A. Ganguly, A. Agrawal, P. Boykin, and R. Figueiredo, "WOW: Self-Organizing Wide Area Overlay Networks of Virtual Workstations" *Proceedings of the Fifteenth International Symposium on High Performance Distributed Computing (HPDC-15)*, 2005.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Building e-Science Portals: A Service Oriented Architecture

Dennis GANNON, Beth PLALE, Marcus CHRISTIE, Yi HUANG, Scott JENSEN, Ning LIU, Suresh MARRU, Sangmi LEE PALLICKARA, Srinath PERERA, Satoshi SHIRISUNA, Yogesh SIMMHAN, Aleksander SLOMINSKI, Yiming SUN, Nithya VIJAYAKUMAR
School of Informatics
Indiana University, Bloomington, IN

Abstract. Grids are built by communities who need a shared cyberinfrastructure to make progress on the critical problems they are currently confronting. An e-science portal is a conventional Web portal that sits on top of a rich collection of web services that allow a community of users access to shared data and application resources without exposing them to the details of Grid computing. In this chapter we describe a service-oriented architecture to support this type of portal.

Keywords. Grid, Portal, Web Services, Service Oriented Architectures.

Introduction

Grid technology has matured to the point where many different communities are actively engaged in building distributed information infrastructures to support discipline specific problem solving. Often called Grids, this distributed infrastructure is based on Web and Web service technology that brings tools and data to the users in ways that enable modalities of collaborative work not previously possible. The vanguard of these communities are focused on specific scientific or engineering disciplines such as weather prediction, geophysics, earthquake engineering, biology and high-energy physics, but Grids can also be used by any community that requires distributed collaboration and the need to share networked resources. For example, a collaboration may be as modest as a handful of people at remote locations that agree to share computers and databases to conduct a study and publish a result. Or a community may be a company that needs different divisions of the organization at remote locations to work together more closely by sharing access to a set of private services organized as a corporate Grid. An eeScience portal is the user-facing access point to such a Grid. It is composed of the tools that the community needs to solve problems. It is usually designed so that users interact with it in terms of their domain discipline and the details of Grid services and distributed computing are hidden from view.

In this chapter we describe a Service Oriented Architecture to support this class portals for eScience collaborations. We base this discussion on our experience with a five-year project [1] to build such a framework for a community of atmospheric

scientists who are engaged in changing the paradigm of severe storm prediction from static "back room" forecasts to an interactive, adaptive data-driven enterprise.

Our concept of the software manifestation that constitutes a "portal" has evolved over the years and, as we look towards the future of Web technology, we can already see signs that it will continue to change. The simplest vision of a "portal" is a desktop application that is an all-in-one appliance for doing domain science in a way that exploits remote data and compute resources.

The most common current model for an e-science portal is based on standard web portal technology such as used by banks, airlines and social networking: a specially designed web server that allows the user to login and see his or her private and group resources. The user accesses such a portal with a standard web browser. As we move to the future, new "Web 2.0" technologies will allow the web browser to be replaced by more interactive, automatically downloaded clients that provide a more seamless integration with the user's desktop.

1. The Components of an e-science portal

There are five common components to most e-science portals.

1. Data search and discovery. The vast majority of scientific collaborations revolve around shared access to discipline specific data collections. Users want to be able to search for data as they would search for web pages using an Internet index. The difference here is that the data is described in terms of metadata and instead of keywords, the search may involve terms from a domain specific ontology and contain range values. The data may be generated by streams from instruments and consequently the query may be satisfied only by means of a monitoring agent.
2. Security. Communities like to protect group and individual data. Grid resource providers like to have an understanding of who is using their resources. This is a specific requirement of Gateway portals such as those that access the TeraGrid. A key feature that distinguishes a portal from a website is the fact that each user has an account. The e-science portal must authenticate each user and associate the appropriate user-specific authorization rights that grant that user access the back-end resource Grid.
3. User private data storage. Because each user must "login" and authenticate with the portal, the portal can provide the user with a personalized view of their data and resources. This is a common feature of all portals in the commercial sector. In the eScience domain private data can consists of a searchable index of experimental result and data gathered from the community resources.
4. Tools for designing and conducting computational experiments. Scientist need to be able to run analysis processes over collected data. Often these analysis processes are single computations and often they are complex composed scenarios of preprocessing, analysis, post processing and visualization. These experimental workflows are often repeated hundreds of times with slightly different parameter settings or input data. A critical feature of any eScience portal is the capability compose workflows, add new computational analysis programs to a catalog of workflow components and a simple way to run the workflows with the results automatically stored in the user's private data space.
5. Tracking data provenance. The key to the scientific method is repeatability of experiments. As we become more data and computationally oriented in our

approach to science, it is important that we support ways to discover exactly how a data set was generated. What were the processes that were involved? What versions of the software were used? What is the quality of the input data? A good eScience gateway should automatically support this data provenance tracking and management so that these questions can be asked.

There are many approaches to supporting this feature set. As previously noted, one obvious solution is to implement all the required capability in a single monolithic application. The application may be a single instance of a portal server, or it may be a desktop application. In fact, our earliest attempts to build science portals followed the monolithic approach and we discovered two major problems: scalability and extensibility. Scalability is a problem with a monolithic portal server. As soon as you have more than a few dozen users logged into a portal server that tries to manage internally all the features described above, it will run out of memory and crash.

To solve this problem, we learned to make the portal server as thin as possible. As illustrated in figure 1, one can build the architecture as a set of specialized services: a service (GFAC) for incorporating new analysis algorithms into the framework, a service for cataloging data products, a workflow manager, a personal metadata catalog, and a provenance service. All of this sits on top of an enterprise service bus that conveys messages between the various components.

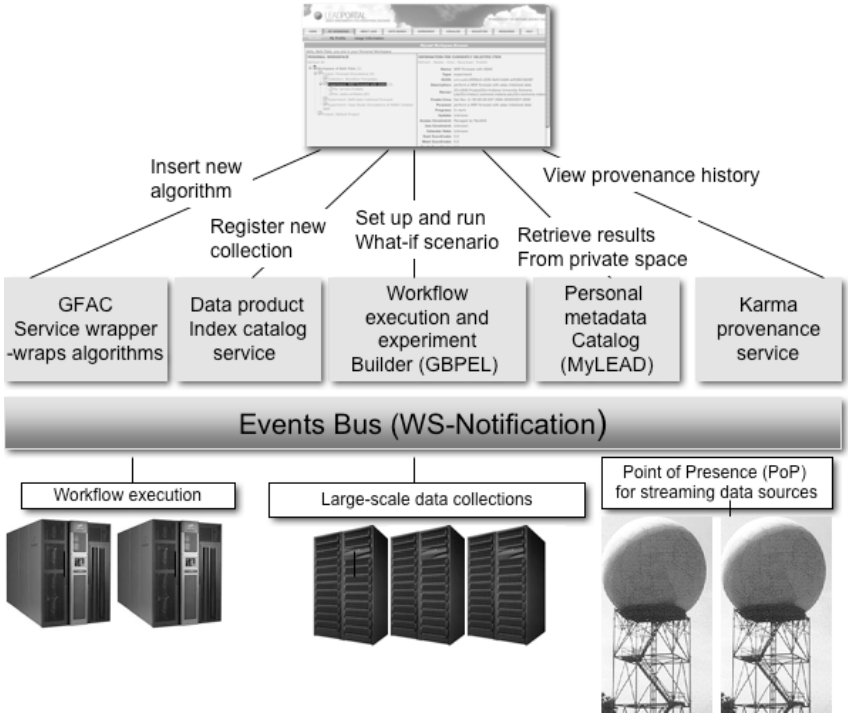


Figure 1. A distributed service-based portal server.

Extensibility means that the framework can easily incorporate new features and capabilities. This is not a strong suite for monolithic application portals unless they are

designed with a strong software component model. However, even with a component architecture, the framework must know how to discover new components and load them automatically. This problem is also related to software maintenance. A change to a client-side application requires that users update the application or periodically install a new version.

A Web portal gives greater flexibility in extending capabilities and doing maintenance. The user only needs to update the browser once or twice a year, while the back end portal server can be continuously updated.

1.1. The Portal Server.

Once the decision to use a web portal framework has been made, there are several issues that must be addressed. First, one must decide on the specific architecture to use. There are many Web portal tool frameworks available from open source and commercial sources. Many are based on a technology called the Java JSR-168 specification [2] in which the portal framework is a container for **portlets**. From the user's point of view a portlet is a windowpane in the browser whose contents defined by the portal manager. From the portal designers point of view, a portlet is a container for an application that manages a user interface. As illustrated in Figure 2, once a user logs into a portlet-based portal they see a pallet with tabs. Each tab, when "clicked" will bring up a page with one or more portlets. Users can add and remove portlets from the collection they see, but most do not bother to do so if the portal manager makes a reasonable selection as the default.

Figure 2. The collection of portlets in the LEAD portal are organized by tabs. Each major tab leads to a second level of portlets. In this case the user has selected "My Workspace" and then the user metadata catalog browser "MyLEAD".

While a portlet can contain a user interface to a substantial application running in the portal server, in our experience, the best design is for each portlet to interface to one of the core Web services that lies behind the portal server. The portlet's job is to provide a user interface for that service. When the user provides input through that interface, a Web service invocation is created by the portlet and sent to the corresponding service. The response is rendered in the user interface. In some cases the portlet is just an interface to a **mashup**, which is a graphical interface that is a combination of information from other services. Our data search interface is an example of one such mashup that we will describe later.

1.1.1. Portal and Grid Security

A major component of any Gateway Portal server is the management of security. Every off-the-shelf portal framework has a way to allow users to request portal accounts and it has the database support for maintaining consistent records of each

user's portal configuration. However, the major challenge for the portal security is to connect this to the security of the back-end Grid. In the standard Globus Grid [3], users are authenticated with the back-end by means of a proxy certificate, which is a temporary public-private key pair that can be used by other agents to act on the user's behalf with distributed Grid services and resources. Given a user's distinguished name and passphrase, a tool called MyProxy [4] can be invoked by the portal server to acquire a Grid proxy certificate for the user. Once the portal server has the proxy, it can pass it to the external web services to be used during interaction with the Grid resources. If each portal user also has a Grid account, the mapping of the user's portal identity to their Grid identity is relatively easy to manage by the portal server.

Unfortunately, most Gateway Portal users do not have a Grid account. They may not even know, or want to know, that the back-end Grid exists. In this case the solution we use in TeraGrid is to give the portal its own Grid identity and account. In this approach, all portal users are mapped to a single Grid account and it is up to the portal services to create an audit trail for each portal user, so the Grid managers can be satisfied that no misuse of the Grid resources occurred by a troublesome portal user. To accomplish this, the portal server must provide additional metadata that is propagated through the system that identifies each portal user and "proof" of his or her authorization to carry the requested Grid operations.

There is much more that can be said on the subject of security, but space does not allow us to go into more detail here.

1.2. A High Level View of the Service Organization

The primary persistent Web services in the LEAD gateway SOA are shown in Figure 3. The data search and discovery portlets in the portal server talk to the Data Catalog Service. As described in the following section, this service is an index of data that is known to the system. The user's personal data is cataloged in the MyLEAD service. MyLEAD is a metadata catalog. The large data files are stored on the back-end Grid resources under management of the Data Management Service. The myLEAD Agent manages the connection between the metadata and the data.

Workflow in this architecture is described in terms of dataflow graphs, were the nodes of the graph represent computations and the edges represent data dependencies. The actual computations are programs that are pre-installed and run on the back-end Grid computing resources. However, the workflow engine, which sequences the execution of each computational task, sees these computations as just more Web services. Unlike the other services described in this section, however, these application services are virtual in that they are created on-demand by an application factory and each application service instance controls the execution of a specific application on a specific computing resource. The application service instances are responsible for fetching the data needed for each invocation of the application, submitting the job to the compute engine, and monitoring the execution of the application. The pattern of behavior is simple. When a user creates or selects an experiment workflow template, the required input data is identified and then bound to create a concrete instance of the workflow. Some of the input data comes from user input and others come from a search of the Data Catalog or the user's MyLEAD space. When the execution begins, the workflow engine sends work requests for specific applications to the Application Factory. If a needed Application Service is not already instantiated, the Factory starts one and the request is sent to the service instance.

A central component of the system is an event notification bus, which is used to convey workflow status and control messages throughout the system. The application service instances generate "event notifications" that provide details about the data being staged, the status of the execution of the application and the location of the final results. The MyLEAD agent listens to this message stream and logs the important events to the user's metadata catalog entry for the experiment. The workflow engine, provenance collection service and data management service all hear these notifications, which also contain valuable metadata about the intermediate and final data products. The Data Management service is responsible for migrating data from the Compute Engine to long-term storage. The provenance collection service records all of this information and organizes it so that data provenance queries and statistics are easily satisfied.

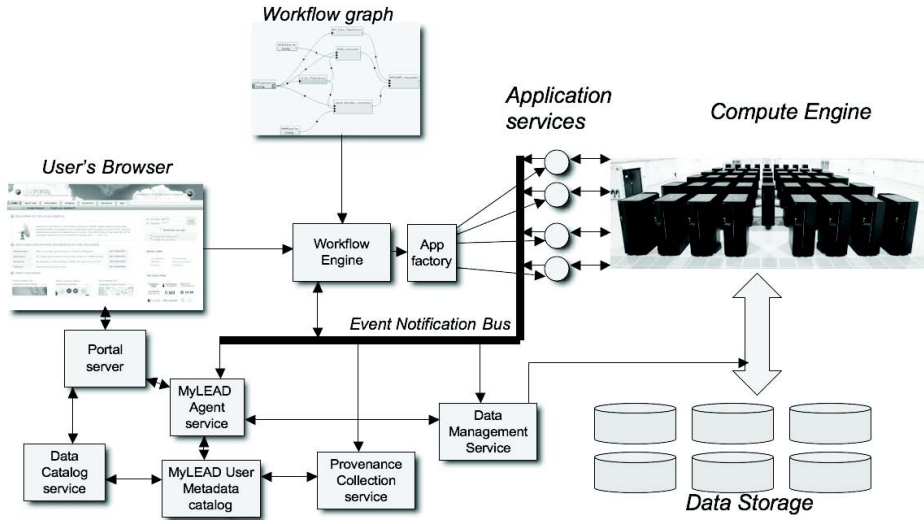


Figure 3. The LEAD persistent services.

In the following sections of this chapter we dig deeper into the role and design of each of these major services.

2. The Data Architecture

Data search, discovery and cataloging are central components of the portal. Most eScience domains have community data collections that serve as the raw information upon which knowledge is built. In the case of atmospheric science, this data comes from a wealth of remote sources including routine aviation reports, upper air soundings from balloon-born instruments, satellite image data, NEXRAD Doppler radar data and various forecast results. Much of this information is distributed through a community supported message-oriented middleware (MOM) layer called the Unidata Internet Data Distribution (IDD) system and catalogued with the Thematic Realtime Environmnetal Distributed Data Service (THREDDS) [5]. Just as a Web search engine crawls websites to build an index, the data subsystem for the portal infrastructure for LEAD crawls the IDD and THREDDS weather data systems and builds a unified index of

these heterogeneous sources. However, crawling and indexing scientific data sources differs from standard Web search engine crawling, largely because of the kind of information crawled and what the user needs to do with the results. A Web search engine need only build an index based on keywords scraped from Web pages. Scientific data is often in a binary representation that is described by metadata. Sometimes the data and metadata are stored separately. The information that must be crawled is often the metadata because that is where we find a semantic description of the contents of a data file. Typical metadata in atmospheric sciences includes the geospatial region it covers and time at which it was generated as well as the generation source (description of the instrument or model that generated it), the binary format of the data, etc.

Scientific metadata is organized according to an XML schema but, unfortunately, a scientific discipline often has multiple competing schemas for describing data. Consequently any data crawler must be able to understand all the pertinent schemas. In the LEAD project a global, extensible schema, called the LEAD Metadata Schema (LMS) [6], was designed so that any of the needed metadata described in the THREDDS catalogs and other important sources could be extracted and cataloged.

Having rich metadata enables far more meaningful data search capabilities than keyword search in a Web search index. In particular, one can build an interface that uses semantic knowledge about the attributes of the data schema. Such an interface can enable much deeper and more meaningful queries. For example, the LEAD data search portlet is shown in Figure 4.

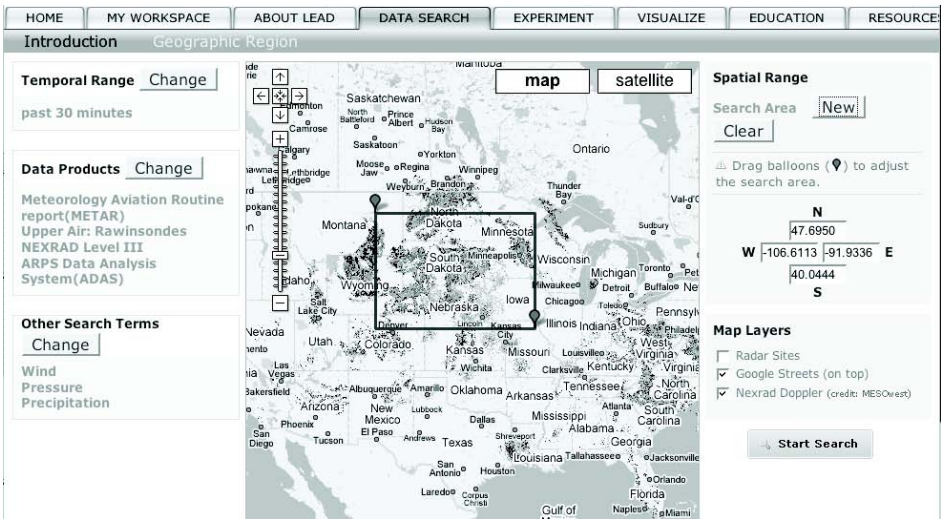


Figure 4. The LEAD data search mashup portlet.

In the figure, the user has constructed a query requesting data produced in the last 30 minutes (upper left) from the Meteorology Aviation report (METAR), the upper air balloon instruments, NEXRAD Level III and the ARPS Data Analysis system data sources. It is further refined to be limited to data about wind, pressure, and precipitation, and to products that contain data about the US prairie states. As can be seen in Figure 4, the current weather radar indicates that stormy weather is currently developing in the US prairie state region. Hitting the "Start Search" button returns a

complete list of all the data that matches the query. For each item in the returned list, the user has the option of downloading it to the desktop, viewing it in a graphics package or saving it to their MyLEAD space for further analysis.

MyLEAD manages the user's personal workspace experience [7,8]. From the user's point of view, it is a browser, as shown in Figure 5, organized as a hierarchical set of folders. For the atmospheric science domain, the top level entities in the workspace are projects. Projects have members called "experiments", and project-wide materials stored to a "collection". An experiment represents a single execution of a analysis workflow. As shown in the illustration, selecting an experiment name in the left hand pane brings up a history of the events that took place during the execution of the experiment. Opening the experiment folder brings up all the metadata records for the input data, intermediate data products and results as well as error reports.

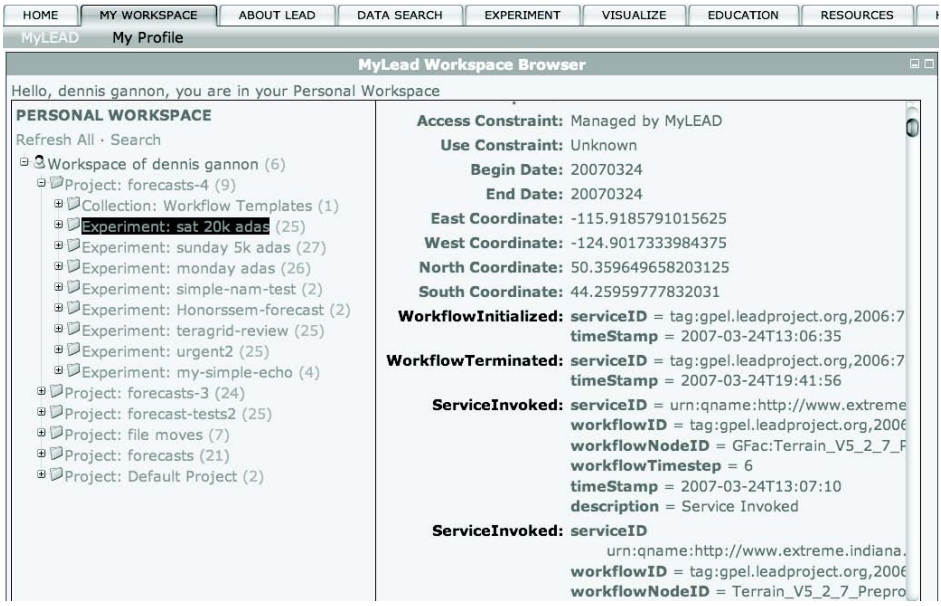


Figure 5. The MyLEAD personal workspace.

Because the MyLEAD workspace conforms to the LEAD metadata schema, it can provide the user a familiar yet highly sophisticated search capability. This search interface allows the user to search over files, experiments and projects, for parents and children, and for any combination of attributes in the metadata schema. For example, one can search for all experiments run on or after April 1, 2007 that used a 2km grid spacing in the simulation and also encountered a fault during execution. This is an extremely powerful feature that is not possible without rich semantic information provided by the metadata schema and the workspace notions of context and organization.

2.1. Data Management.

Data transfer, replica management, and naming are related functionalities required of a storage service in a cyberinfrastructure. While third-party data transfer allows clients to move data from one host to another without having to itself monitor the

transfer, replica management allows optimal selection of replicas when transferring the data. Naming provides location transparency by using logical identifiers for the files and mapping them to the physical locations of the replicas Data movement, replication, and naming is provided by the Globus Toolkit DRS, and by the simpler Data Movement and Naming service that was inspired by DRS. The Data Movement and Naming (DaMN) service, for instance, provides support for data transfers through simple GET and PUT service operations. The GET operation to transfer data out of a data repository takes a logical name for the source data and a logical or physical destination where the data needs to be transferred to. As a precursor to the transfer, DaMN uses the Replica Location Service (RLS) [9] to resolve the logical data name to the physical locations of the data products, optionally using algorithms for optimal replica selection. DaMN separately maintains a list of logical myLEAD repository locations to use as destination locations and resolves them to pre-configured physical storage locations. The capabilities of Reliable File Transfer (RFT) [10], based on GridFTP, is leveraged to effect third-party data transfers between the resolved source physical location and the destination physical location in a secure and reliable manner. After the transfer is complete, the new replica's location is registered with RLS, ensuring accurate tracking of replicas. The DaMN service provides polling, callback, and notifications as means of notifying the client of the successful data transfer and replica registration.

2.2. Data Provenance: The key to the modern scientific method.

Provenance is a form of metadata that describes the derivation history of data products derived from data transformation pipelines [11]. The Karma provenance framework [12] collects runtime information from scientific workflows and the services that constitute them. Karma defines an information model and messages that are generated by instrumented services and workflows at control transfer boundaries and data creation and usage points in the workflow execution. These messages are pushed from the instrumentation points to the Karma service through the portal's publish-subscribe system. From the information it receives, the provenance service can construct a global causal graph of the data derivation trail that can span across individual workflows when data is shared between them. Service APIs to query for different forms of provenance graphs [13] are exposed and graphical clients that can view complete provenance graphs or monitor the realtime provenance activities are available.

The provenance system supports queries of the form "Find the process that led to Atlas X Graphic, excluding everything prior to the averaging of images with softmean" or "Find all invocations of procedure align warp using a twelfth order nonlinear 1365 parameter model that ran on a Monday" are possible.

3. Turning Scientific Applications into Web Services

The concept of application-as-a-service is one that has gained considerable interest in the commercial software world. In this model a service provider is responsible for maintaining, updating and executing the application on-demand for clients. The client need only access the application interface through a Web service. However, this has not been widely used in the eScience domain until recently. In the LEAD project we have

number of stable "community codes" which can be combined in various ways to do data analysis or to run complex weather forecast scenarios. The problem is that these are Fortran codes that are executed as command-line programs. To make them composable, we turned them into virtual Web services. The services are virtual for a simple reason. If we had a persistent Web service for each science application on each supercomputer, we would have dozens of services to keep alive. To solve this problem we have a single persistent factory service capable of creating an instance of a needed application service on-demand. The specific application service instance need only persist as long as it has work to do. Because the process of creating a virtual service from a command-line program is so simple, we have automated it as a service available through the portal known as the Generic Application Factory GFAC [14].

The portal guides the user through the creation of two documents.

1. The service registration document contains the service name, the executable path for the application, the input parameters names and types and types, the output parameter names and types and details about the execution parameters such as the number of processors to use when running the application.
2. A host description document, which contains the details about the host where the GFAC service has been deployed and the application services will run. (This document is only needed if the host is not previously known to the portal. In the case of a Grid deployment, the host on which the service instance is running need not be the same as the host upon which the application runs. Globus GRAM is used to remotely invoke the application.)

With this information, the GFAC service generates a standard Web Service Description Language (WDL) document and can instantiate the service on-demand.

When an application service is invoked with a given set of input parameters, it runs the application with those input parameters (possibly on a cluster of resources), monitors the application and returns the results to the user. For debugging GFAC also provides a generic web service client that allows users to securely access any application service created by the toolkit from the convenience of a portal. When a user accesses an application service, the user is presented with a graphical user interface (GUI) to that service. The GUI contains a list of operations that the user is allowed to invoke on that service. After choosing an operation, the user is presented with a GUI for that operation, which allows the user to specify all the input parameters to that operation. The user can then invoke the operation on the service and get the output results.

4. Composing service: Workflows and Exploration Patterns.

Scientists conduct experiments and the portal is designed to support that. As we have seen, the MyLEAD personal metadata catalog is organized around this concept. We have identified two ways users may wish to conduct computational experiments. The first of these we refer to as task-driven, in which the user wants to run a particular type of data analysis or simulation and then gathers the needed data, and the other is data-driven, where the user wants to search and discover data and then find the tools that can be applied to it. In this chapter we address the former model, but the later is one we are currently investigating.

The way an experiment is initiated in the portal is through the Experiment Builder. Based on the idea of a software "wizard", a program which leads the user through the steps for installing or using software, the experiment builder guides the user through the setup and execution of an analysis workflow. As illustrated in Figure 6, the first thing the experiment builder asks the user is to identify the project that the experiment belongs to, the name of the experiment and a brief description.

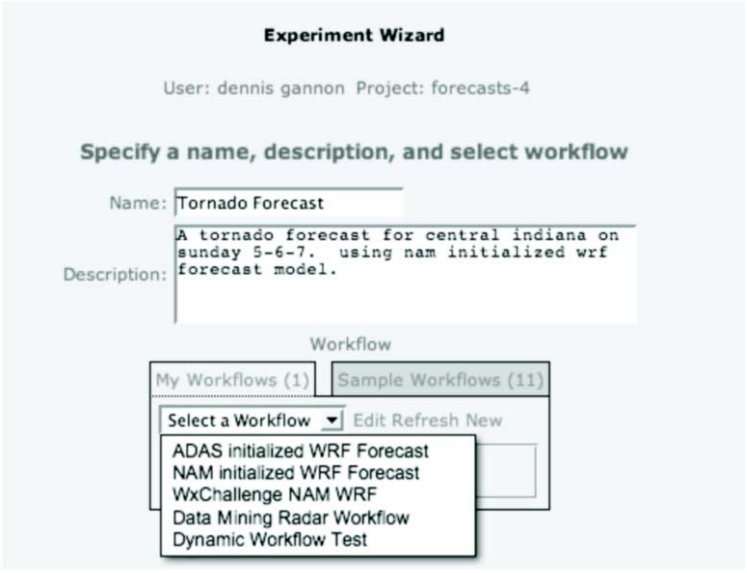


Figure 6. The Experiment Builder Wizard.

Next the user selects a workflow from a list of predefined templates. The user may edit the template or create a new template using the workflow graphical editor. The workflows are represented as a combination of dataflow and control flow. Most scientists are comfortable with the concept of dataflow represented as a graph. Each node of the graph represents an application service with multiple inputs and one or more outputs. The inputs and outputs may be as simple as a numerical or string value, or as complex as a descriptor of a set of files, which must be fetched and staged on the compute host that is running the application. (This staging is done automatically by the application service instance.) An example is illustrated in Figure 7 from the LEAD project. In this case the application service is a data interpolator that produces lateral boundary conditions for a forecast simulation. It take three input file descriptors. One describes the terrain of the forecast domain. The second is a file containing a set of parameters that describe the domain and the mesh spacing for the computation. The third is data that comes from the National Centers for Environmental Prediction (NCEP) North American Mesoscale (NAM) forecast. The output is an interpolated lateral boundary condition that is used as part of a storm forecast.

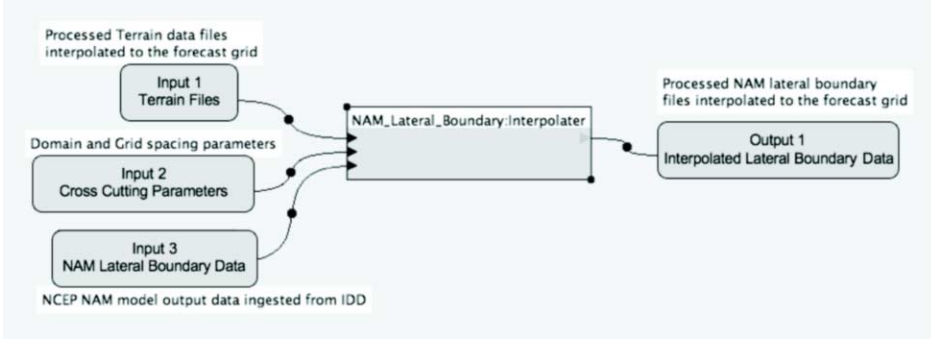


Figure 7. An application service from a weather forecast workflow. The service is an interpolator for the lateral boundary conditions for the forecast simulation domain.

To compose a complete workflow, the portal user can invoke a tool called XBaya that provides a graphical interface to build the workflow. Shown in Figure 8, XBaya is a conventional "drop and drag" graph composer. The list of available workflow components and application services are shown on the left. Each of these components can be placed on the composition pallet to the right and wired into the workflow, which is shown add a left-to-right directed acyclic graph.

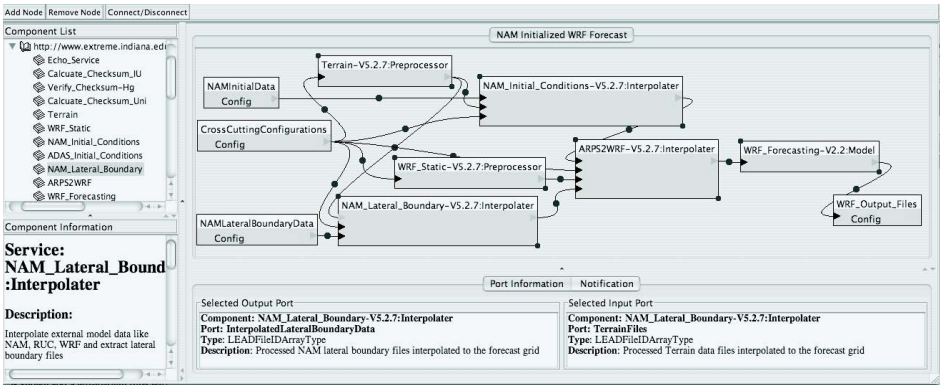


Figure 8. The XBaya workflow composer and monitoring tool.

The workflow shown in Figure 8 is a LEAD project storm forecast simulation that takes inputs consisting of a set of configuration property files that initialize parameters for different parts of the weather simulation and a set of North American Model (NAM) data files that define the initial conditions of the forecast.

XBaya is a compiler which translates this graphical description into a lower level workflow language. Because XBaya is so high level it is possible to map the graphical workflow description into many different workflow language and two are supported now. For long running workflows, BPEL [15] is the most completely supported Web service workflow language. If the workflow is short running, then the user may wish to

execute the workflow directly from their desktop machine. In this case, XBaya can generate the Java dialect of Python, called Jython.

When the user builds a workflow with XBaya, he or she is actually building an abstract workflow template. What is missing is the specific input data and the specific instances of the application web services. That binding is left to execution time for two reasons. First, in the case of workflows that must consume data that is timely (such as weather data), one does not want to bind the data until the workflow is almost ready to run. Second, because the applications runs on many different host computers, it is best to select the application services instance that corresponds to a host that has the lightest load or greatest capacity. In this case, the selection may be deferred until just prior to invoking the abstract service. The way this is done is the actual service request is sent to the application factory service, which through an intermediate service, makes a call-out to a resource broker that determines the best choice for a selected instance of the application service. When this best service is known the factory forwards the Web service there and the results return to the workflow engine.

4.1. The Workflow Engine.

There are a large number of possible workflow systems that can be and are being used in eScience projects and portals. In the case of Web service based frameworks, we have found that the BPEL standard is well suited to our needs for several reasons. BPEL is designed for long running workflows that are common in eScience applications that must use services that are widely distributed over a Grid. We have implemented a version of the standard we call GPEL [16] with several special features that are significant for applications like LEAD. Specifically, we require that

1. A workflow may run for hours or, in the case where the workflow is waiting for significant external events such as storm warnings to trigger actions, the workflow may run for days or weeks. Consequently the workflow instance state must be saved in stable storage.
2. The workflow must be able to respond to dynamically changing conditions including crashes of parts of the Grid or massive new demand created by a scenario, such as a major storm, or even the desire of the scientist to interrupt it and modify it or redirect it.
3. In case additional requirements are needed by the scientist, it may be necessary for a workflow to be restarted or cloned from an intermediate state.

In GPEL the state of every workflow instance is represented by an XML document stored in a database. A change of state, such as advancing the workflow to a new state because of the completion of some action, is just a transformation of that document. Consequently the three conditions listed above are easily met. One capability that we did not plan on was the ability to restart workflows that failed because of back-end Grid failures. Because the workflow state and all intermediate data products are saved by the data system, it proved to be a simple task to retrieve the state of a failed workflow and "resurrect" it using different resources.

One question that remained was scalability. While in production the system was able to manage over 1200 workflows a month and in the same time the data system managed over 2.6 terabytes of data product. A more detailed study of the performance of the entire system will be conducted over the year ahead.

4.2. The Pub-Sub Service Bus.

Web services are designed to receive messages and act upon them in the most stateless manner possible. However, if every service must rely upon other services to directly engage them for every piece of vital information, the architecture becomes very brittle and will not scale. The standard approach to solving this problem is to allow services to respond to events that represent changes to the state of workflow instances or Grid deployment status or external demands on the system. To address this problem most scalable distributed systems rely on a "publish-subscribe" event backbone. The idea is very simple. Persistent services such as the data subsystem or the providence tracking system that must know when certain thing happen, such as when a file has been moved, or an application service has terminated or encountered an error can do so by creating a subscription to an event service that will notify them by a simple message when something important happens. A subscription consists of a message to the event service indicating that a particular "listener" service is interested in receiving an invocation anytime a message is "published" on a particular topic. A topic may be any event associated with a specific user's experiment workflow state transition or an application service status message. Or it may be a message indicating that the Grid has undergone a subsystem failure. It may also be a notification that a severe storm has been detected.

There are two main Web service standard for pub-sub event systems: Ws-Eventing and Ws-Notification. The underlying Grid infrastructure, if based on Globus, uses WS-Notification, but a more widely adopted standard is WS-Eventing. Our approach has been to mediate between these two systems. Our system is based on a tool we call WS-messenger [17] that supports both standards and is an internet-scale reliable publish-subscribe system. This system allows both services and user client programs to always maintain a current picture of the state of each workflow. For example, the data providence and MyLead broker services rely on these notifications. XBaya, is able to track the state of a user's workflow by subscribing to the event stream. As the state of the workflow changes the graphical view of the workflow can show the user the current status. This has proved to be an invaluable in debugging and the users find the feedback about experiment status to be extremely helpful. The system has proven to be extremely reliable, but a quantitative study has not yet been completed.

5. Responding to the external world.

As described above, an event system is essential for monitoring the state of the service architecture and the progress of workflows. However, the external world generates events that are of great interest to scientists.

Atmospheric scientists have since the beginning of the LEAD project wanted the e-science portal to give them a way to launch a regional weather forecast model in response to developing severe storms. In order to provide this, the portal has been extended with support for ingesting model- or radar-generated data in real time, for mining and filtering the streaming data using continuous query processing to filter and aggregate data [18] and through software classification algorithms that can continuously mine the results [19]. When the classification algorithms detect severe storm signatures exceeding a threshold, a notification message is sent.

As an example, suppose a user wishes to keep an eye on the storm front moving into Chicago later that evening. He/she logs into the portal, and configures an agent to observe NEXRAD Level II radar data streams for the next 4 hours looking for severe weather developing over the Chicago region. The request is in the form of a continuous query that executes the mining algorithm repeatedly. The query execution service will subscribe on-the-fly to streams of observational data. The events arrive as bzipped binary data chunks and are broken open to extract metadata. The resulting XML activities are subject to filtering and aggregation and the zipped product passed to the classification tool. Data mining will result in a response trigger, such as ``*concentration of high reflectivity found centered at lat=x, lon=y*'', that is sent to the workflow engine using the WS-Eventing notification system. WS-Eventing uses an internal event channel for the transfer of data streams.

6. Related Work.

The architecture we describe here was not created in a vacuum. There has been a community engaged in the design of e-science portals for the last five years and a substantial number of contributions have been made. The have been a long line of e-science portals aimed at managing the basic user interaction with a Grid including GridPort [20] and GENIUS [21]. The Gridsphere project [22] was an early pioneer that moved to bring the JSR-168 and the Gridsphere portal framework has been the most widely used of the portlet containers for eScience applications including the one described here.

There have been dozens of eScience portals built for different communities and scientific disciplines. In the domain of biology there are several including the RENCI BioPortal [23]. The GEON Portal [24] is a gateway to geosciences data and applications. The BIRN Portal [25, 26] is the leading eScience gateway for biomedical imaging. The Cactus Portal [27] is a front end for Grid tools for the study of black holes. The area of earthquake science is covered by Servogrid portal [28] and the NEES Grid portal [29] was built for earthquake engineers to share access to specialized instruments. NanoHub [30] is an e-science portal for nanotechnology and nanoscience and it is the most heavily used and successful of all e-science portals.

The National Science Foundation TeraGrid Project is now the leading organization in eScience portals. Their TeraGrid Gateways program [31] now has over 20 e-science portals for various scientific disciplines that are accessing the compute and data resources of TeraGrid.

7. Acknowledgements

The work described here on the LEAD portal has had the benefit of substantial design, evaluation, testing and encouragement of many people. Notable contributions came from Kelvin Droegemeier and Dan Weber and the rest of the team from the Center for Analysis and Prediction of Storms at the University of Okahoma and the other LEAD project principal investigators including Dan Reed, Sara Graves, Mohan Ramamurthy, Sepi Yalda, Richard Clark, Everette Joseph and Bob Wilhelmson. Katherine Lawrence and Il-Hwan Kim provided invaluable user interface evaluation and design. The testing of the portal with real users would not have been possible without the help of Richard

Clark and Tom Baltzer. The LEAD metadata schema was created by a team that included Rahul Ramachandran and Anne Wilson. Gopi Kadaswamy was critical in the GFAC design and current work on fault tolerance with Dan Reed. This work was supported by NSF awards ATM 0548692, ATM 0648857, OCI 0503697, and CNS 0330613.

References

[1] B. Plale, D. Gannon, J. Brotzge, K. Droegemeier, J. Kurose, D.McLaughlin, R. Wilhelmson, S. Graves, M. Ramamurthy, R. Clark, S. Yalda, D. Reed, E. Joseph, V. Chandrasekar: CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting. *IEEE Computer* 39(11): 56-64 (2006)

[2] Java Community Process, JSR 168: Portlet Specification, http://jcp.org/en/jsr/detail?id=168

[3] The Globus Alliance. www.globus.org/

[4] J. Novotny, S. Tuecke, V. Welch, An Online Credential Repository for the Grid: MyProxy, *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01)* p. 104, 2001.

[5] Unidata http://www.unidata.ucar.edu

[6] B. Plale, R. Ramachandran, and S. Tanner, Data Management Support for Adaptive Analysis and Prediction of the Atmosphere in LEAD, *22nd Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology (IIPS)*, January 2006.

[7] Y. Sun, S. Jensen, S. Lee Pallickara, Personal Workspace for Large-scale Data-driven Computational Experimentation. *International Conference on Grid Computing (Grid'-6)*, 2006.

[8] B. Plale, D. Gannon, J. Alameda, B. Wilhelmson, S. Hampton, A. Rossi, and K. Droegemeier Active Management of Scientific Data *IEEE Internet Computing* special issue on Internet Access to Scientific Data, Vol. 9, No. 1, Jan/Feb 2005, pp. 27-34.

[9] Globus, Replica Location Service (RLS), http://www.globus.org/rls/

[10] Globus Toolkit version 4.0. Reliable File Transfer (RFT), http://www.globus.org/toolkit/docs/4.0/data/rft/

[11] Y. Simmhan, B. Plale, D. Gannon, A Survey of Data Provenance in Grids. *SIGMOD Record* (Special Section on Scientific Workflows), Vol. 34, No. 3, pp. 31-36, 2005

[12] Y. Simmhan, B. Plale, D. Gannon, A Framework for Collecting Provenance in Data-Centric Scientific Workflows. *International Conference on Web Services (ICWS)*, 2006

[13] Y. Simmhan, B. Plale, D. Gannon,. Querying capabilities of the karma provenance framework. *Concurrency and Computation: Practice and Experience*, 2007.

[14] G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon. Building Web Services for Scientific Grid Applications. *IBM Journal of Research and Development*, 50(2/3):249-260, 2006

[15] Business Process Execution Language Specification Version 1.1. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[16] A. Slominski, Adapting BPEL to Scientific Workflows, Chapter 14 in *Workflows for e-science*, Tayler, Deelman, Gannon, Shields, eds. Springer, 2007.

[17] Y. Huang, D. Gannon, A Flexible and Efficient Approach to Reconcile Different Web Services-based Event Notification Specifications, *International Conference on Web Services (ICWS)*, Chicago, 2006. pp. 735-742.

[18] Y. Liu, N. Vijayakumar, and B. Plale, Stream Processing in Data-driven Computational Science *7th IEEE/ACM International Conference on Grid Computing (Grid'06)*, Barcelona, September 2006.

[19] T. Hinke, J. Rushing, H. Ranganath and S. Graves.. Target-Independent Mining for Scientific Data. in *Proceedings: The Third International Conference of Knowledge Discovery & Data Mining*. 1997.

[20] M. Dahan, et. al., Build e-science portals with Grid portal Toolkit 3, 19 Oct. 2004. http://www-128.ibm.com/developerworks/grid/library/gr-gridport/

[21] A. Andronicoa, R. Barbera, A. Falzonec, G. Lo Rea, A. Pulvirentia, and A. Rodolicoc, GENIUS: a web portal for the grid, Nuclear Instruments and Methods in *Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* Volume 502, Issues 2-3, 21 April 2003, Pages 433-436.

[22] J. Novotny, M. Russell, O. Wehrens, *Concurrency and Computation: Practice and Experience*. Volume 16, Issue 5 , Pages 503 – 513, Mar 2004.

[23] A. Blatecky, K. Gamiel, L. Ramakrishnan, D. Reed, and M. Reed. Building the Bioscience Gateway. In *Science Gateways: Common Community Interfaces to Grid Resources* Workshop at Global Grid Forum 14 (GGF14), June 2005.

[24] U. Nambiar, B. Ludaescher, K. Lin, C. Baru, The GEON portal: accelerating knowledge discovery in the geosciences, Workshop On Web Information And Data Management archive. *Proceedings of the eighth ACM international workshop on Web information and data management*., 2006.

[25] A. Lin, et al., The Telescience Project: Applications to Middleware Interaction Components (2005) Proceedings of The 18th *IEEE International Symposium on Computer-Based Medical Systems*, p. 543 -548.

[26] S. Peltier, A. Lin, D. Lee, S. Mock, S. Lamont, T. Molina, M. Wong, M. Martone, M. Ellisman, The Telescience Portal for Advanced Tomography Applications. *Journal of Parallel and Distributed Applications*, Special Edition on Computational Grids 63(5): 539 – 550. 2003.

[27] I. Kelley, O. Wehrens, M. Russell, J. Novotny. The Cactus Portal. In proceedings of *APAC 05: Advanced Computing, Grid Applications and eResearch*, (2005).

[28] M. Pierce, M. Aktas, G. Aydin, G. Fox, H. Gadgil, Simulating Earthquakes: the QuakeSim/SERVOGrid Project, Chapter 7 in *grid portals*, Gannon, Pierce, Plale eds., to appear 2008.

[29] C. Severance, T. Haupt, NEES - Network for Earthquake Engineering and Simulation, Chapter 2 in *Grid Portals*, Gannon, Pierce, Plale, eds. 2008.

[30] M. McLennan, R. Kennell, D. Ebert, G. Klimeck, W. Qiao, Hub-based Simulation and Graphics Hardware Accelerated Visualization for

Nanotechnology Applications, IEEE Transactions on Visualization and Computer Graphics archive Volume 12 , Issue 5 (September 2006) pp. 1061-1068, 2006.

[31] TeraGrid. http://www.teragrid.org/programs/sci_gateways/

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

A new resource brokering and scheduling solution for a Grid environment

Lucio GRANDINETTI [a], Maria Carmen INCUTTI [a], Francesco MARI [b]
[a] *Università degli Studi della Calabria, Via P. Bucci 41C, Rende (CS) 87036, Italy*
[b] *CESIC-NEC, c/o Università degli Studi della Calabria,*
Via P. Bucci 22B, Rende (CS), 87036, Italy

Abstract. The problem of scheduling a parallel application on to heterogeneous distributed computing environment is one of the most challenging problems in a grid. Two main problems arise: how to efficiently discover resources capable of providing the demanded computing services and how to schedule them. We propose a new fully decentralized solution for both resource brokering and scheduling in a grid based on a peer-to-peer model, which consider resource brokering and scheduling as parts of a single activity, in order to achieve high resources utilization and low response times. The proposed scheduling solution uses the relationship among neighbours to let each node learning structural and functional information about the resource-sharing environment. This mechanism makes the system an intelligent organism, characterized by the capability of self-learning about its environment.

Keywords. Grid Scheduling; Resource Brokering and Management.

Introduction

The promise of grid computing is to link heterogeneous resources, shared among different organizations and geographies, into one virtual resource which can be used by global enterprises and provided as a service on demand.

A computational grid [1][2][3][4] (or metacomputer) is an hardware and software environment based on open standards and protocols. It gives the possibility to use different, freely/not freely coupled Information Technology (IT) resources owned by multiple organizations located all over the world. Built on pervasive Internet standards [5], grid computing allows research-oriented organizations to solve problems that computing and data-integration constraints turn into infeasible; it also reduces costs through automation, improving at the same time IT resource utilization. As a result, this increases organization's agilities permitting more efficient business processes. The challenge is to develop technologies, standards and solutions in order to make grid computing a powerful strategy for the grid environment.

A grid is not a ready-made solution, but rather a set of protocols and components mixed together to build efficient solutions. To achieve this goal, an efficient grid scheduling system [6] is an essential part of a computational grid. Nowadays, this is particularly true: with the success of e-commerce and the Internet, computational grids have been moving from scientific and technical research to IT model based on Web

services [7], in which software and resources can be offered and consumed as services. Grids have the potential to address large scale problems, but how to efficiently and effectively utilize their resources is solved only through an efficient job scheduling method.

Grid scheduling is the process of efficiently allocating grid resources to applications. The grid scheduling system is required to manage the job requests. For each user request, the grid scheduling task includes the research of suitable resources and, after that, the coordination of the job executions. It generates the job schedules for all computational machines by taking into account static and dynamic parameters, with the objective of maximizing the resources utilization.

In 1987, Catlett and Smarr introduced the concept of connecting computing resources establishing the term "*metacomputing*". Since that time many research projects (Globus [8][9][10] and Condor [11][12] are probably the most successful examples) showed that the topic of job scheduling is an important step for constructing efficient grid computing infrastructures and definitely underlined that the job scheduling in a metacomputer significantly differs from job scheduling on a single parallel computer.

A computational grid scheduler has to take into consideration local and global objectives.

The local objectives are concerned with the administrative policy of each site. Typically the key objectives of a local scheduler are: maximizing the overall system performance (high throughput and high resource utilization rate overcoming heterogeneousness of computing resources); supporting various computing intensive applications (batch jobs and parallel applications); providing robust remote job execution functionalities (reliable remote I/O and efficient job management involving job preemption, migration, and checkpointing).

While the scheduler of a computing site can coordinate the execution of the submitted jobs without taking into consideration external restrictions, a scheduling infrastructure in a metacomputing system is subjected to various constraints: the geographical distribution of the machines; the different owner organizations (each one characterized by a different local scheduling policy); the network resources. Moreover, the scheduler has to take into account additional requirements for security, fault tolerance and frequent resource changes, as each single machine may join or exit the grid at any time. In a dynamic distributed computing environment resource availability may vary significantly. Therefore, the grid scheduling becomes a high challenging task.

The metacomputing scheduling policy significantly depends on the structures that occur in computational grids.

The most common architecture in metacomputing environment is based on centralized models. In a centralized scheduling architecture all jobs are submitted to a single grid scheduler which acts as a central broker, responsible for making global decisions and assigning each job to a specific machine maintaining up-to-date information on the state of all available resources. The grid scheduler works as "*access point"* to the whole infrastructure. When a job arrives, the grid scheduler calculates its computational times (TC) for all sites; selects the site that guarantees the minimum TC value; and sends the submitted job to it. Therefore, the scheduler is in charge of directly controlling the resource interfaces and straight interacting with the local resource managers. In a centralized environment the scheduler is conceptually capable of elaborating very efficient schedules, since it has all the necessary information on the

submitted jobs and the available resources. The quantity of information that need to be evaluated does not scale well with the increasing size of the metacomputer. Therefore the central scheduler often may result in a performance bottleneck for large-scale grid environments. As a result, a centralized architecture might be a good choice for a relatively small set of computing resources or for a computing centre where all resources are used under the same objective.

Opposite to the centralized scheduling architecture is the decentralized one, that better matches with the distributed nature of the grid.

In decentralized systems there is neither a single access point to the whole infrastructure nor a central entity responsible for the job scheduling. Distributed schedulers interact one with each other and commit every job to the site that best guarantees to execute it. All the information about jobs and available resources are not evaluated every time by a single entity. This means that the decentralized architecture might lead to sub-optimal scheduling solutions but, at the same time, this eliminates the communication bottleneck of a centralized structure, providing better scalability, reliability and fault tolerance. The communication between local schedulers can be managed in a direct or indirect way. In the first scenario the local schedulers can directly send jobs to other schedulers: each local scheduler receives identical jobs. If it is not possible to immediately schedule a job, it searches for another machine where an immediate scheduling is possible. If a site is found, the job and all its data are transferred to the selected machine. In an indirect communication policy jobs, which cannot be immediately executed, are sent to a central job pool instead of a different site. In this scenario, when a local scheduler cannot immediately execute a job, it pushes it into the pool; at the same time if a site has not any job to schedule, it can pull jobs out of the pool.

Today the majority of grid systems are based on centralized grid scheduling structures. In the future, when the grid will be supposed to provide computational power on demand to all users without requiring any other information, this organization will not be feasible. The technological progress during the last years let us think that this future is not so far. Therefore, nowadays peer-to-peer technologies [13] and totally decentralized scheduling protocols are emerging.

Peer-to-peer systems are Internet applications using resources provided by a number of autonomous participants. The peer-to-peer communities exhibit remarkable sharing patterns, such as free riding and intermittent resource participation. Of course, the research for suitable resources and their coordination in a peer-to-peer system is completely different.

We propose a new fully decentralized scheduling solution for a computational grid environment based on a peer-to-peer model. According to this model each node is connected to other nodes; the proposed solution uses the relationship of each site with its neighbours to learn structural and functional information about the resource-sharing environment. It gives the possibility to automatically learn information regarding resource composition, availability and status. The projected decentralized scheduling solution provides scalability, reliability and fault tolerance.

The chapter is structured as follows. In Section 2, we give a description of the background of our work. Our resource discovery and scheduling solution is presented in Section 3. In Section 4, we show the results of our preliminary tests. Finally, we conclude and we give an overview of future work.

1. Background

The workload of a grid is a set of jobs submitted by independent users.

The scheduling system has to decide when and where each job can be executed.

The complexity of an efficient and flexible grid scheduling system is very high, but from the functionality point of view, it is still possible to find a logical structure of the scheduling process in a grid [14].

A grid scheduling process can be generalized into two stages: the resource brokering and filtering, and the resource mapping.

The resource discovery is an essential service in any resource-sharing environment: given a description of the desired resources, a resource discovery system returns the locations of resources matching the evaluated description. In order to be efficient in a distributed environment, this module can be implemented on an information manager module. It is responsible for collecting and distributing information about resource availability: the quality of these data may have a strong impact on scheduling algorithms and overall performance.

Another essential service in any resource-sharing environment is the resource mapping. This service is concerned with the coordination of the sharing resources matching each evaluated application description. It is a very complicated problem to solve because a grid is an heterogeneous system, and the scheduling component generally does not enforce absolute control over the resources.

The complexity of resource discovery and allocation problem, that definitely characterizes a grid scheduling strategy, strongly depends on the size of a grid and the diversity of the access and scheduling policy of each administrative provider.

To design and evaluate different grid scheduling strategies it is necessary to distinguish two grid computing setting environments: High-Performance Computational grids (HPC grids), and World-Wide grid.

While the first grid environment currently represents the use case of a grid, the second one represents the long-term vision for grid technology.

A HPC grid is made up of a limited number of computing sites, each one characterized by high performance computer systems, strictly interconnected by a high performance network that allowed their cooperation. Normally to each HPC grid is associated a dedicated user group, that generally is a research and educational community in which sometimes it is possible to find large commercial companies working on important research and simulation activities. All around the world there are many HPC grid, generally supported by national ad intra national investments. Some computational grids are based on resources owned by a single administrative domain, for instance an important University or a large company, but most commonly a HPC grid consists of resources with different providers. In this case, a specific grid management system is needed, but an appropriate scheduling architecture has not been established yet. The problem of finding apposite resources for each request is not very difficult because a HPC grid is characterized by a limited number of computing sites in which the different providers perfectly know all the others. Also the decision about the best available resources on which a submitted application has to be processed is not a very hard decision to make; in this scenario the user's applications are massively computing applications characterized by processing time values between some hours to some days, so generally the used scheduling policy is based on a First In First Served with priority procedure. The management of the applications priorities depends on the environment in which it is performed; in the centralized environment the global grid

scheduler makes global decisions; in the decentralized environment each local scheduler decides the priorities basing on particular objectives.

A World-Wide grid is a large-scale grid with independent users and a large number of resources generally belonging to independent and typically unknown providers. In the World-Wide grid the complexity of the scheduling task is completely different. In the next future, the World-Wide grid will use the resources of million of separate computing sites connected by a network (usually the Internet) to solve large-scale research problems as well as business one. The World-Wide grid will have the capability to perform computations on large data sets, by breaking them down into many smaller ones, and to manage many computations at once, by providing a parallel division of the workload between processes distributed all over the world. A user will see the grid as a single huge and powerful computer system, providing almost infinite computational power. The user will enter the grid and ask to execute a job using a software interface that will run on his computer. After clearing security validation, his computer will be able to connect to a Grid Resource Broker module [15] which will perform resource discovery, scheduling, and the processing of application jobs on the distributed grid resources. In particular, to find the apposite resources for each request the Resource Broker module will query the Information Service module [16] to know which resources are feasible and among them which are available at that moment, and also to have information about the locations of all existing data. Then, the Resource Broker module will find the best resources to execute the submitted job anywhere on the grid. When the appropriate resources will be chosen the job can be executed, and the Resource Broker will send the results to the user. On the other hand, finding apposite resources for each request is a very hard task, because this computing scenario is characterized by a huge number of computing sites in which the different providers do not know all the others and the dynamic nature of the grid requires adaptable solutions. With so many resources in the grid, the probability of some resource failing is high, so resource failures can be considered the rule rather than the exception.

The decision about the best available resources on which every submitted application has to be processed is not easy; in this scenario the user's applications are not only massively computing applications characterized by heavy processing time. Therefore, the scheduling policy becomes more important since there is the necessity to offer to the user a high service level taking into consideration several structural aspects of the World-Wide grid environment [17]: heterogeneity; scalability and reliability; site autonomy; online problem; co-allocation; resource reservation.

Heterogeneity. The World-Wide grid compounds a large variety of resources that are heterogeneous in nature and involves a huge range of technologies.

Scalability. The World-Wide grid might grow from a few integrated resources to millions. This stirs up a problem: the potential performance degradation as the grid increases. As a result, applications requiring a large number of geographically located resources must be designed to be latency and bandwidth tolerant.

Site autonomy All resources of a grid are typically not owned and maintained by the same administrative instance. The autonomy of resource owners, in particular of independent schedulers, regards heterogeneous substrate, variable scheduling objectives, and so on. A grid scheduler has no exclusive control over all resources in a grid. Therefore, it must cooperate with the existing independent local schedulers. The resources usually have their own local management software with different features depending on the local policies and functionalities of the management software. Hence, the grid scheduler has to cope with the limitations of the local management. The

scheduling objectives may vary for each available resource according to the provider's policy. In addition to the objectives of the provider, the users' needs must also be taken into account.

Online problem The grid scheduling takes place in an online environment where jobs are submitted at any time. Additionally, information on the current state of resources may be difficult to obtain and to keep up to date.

Co-allocation Some applications need several resources from different providers at the same time. The grid scheduling system must be capable of coordinating these resources.

Resource reservation Several complex applications require the reservation of resources in advance. Further, it is advantageous for the scheduler to consider system downtime or restricted access that is known beforehand. Finally, advance reservations are required for co-allocated resources or multi-site applications.

The focus of this chapter is the description of a new solution for both resource brokering and scheduling in a World-Wide grid based on a peer-to-peer model. This grid environment, characterized by highly variable resource and user participation, requires fully decentralized solutions that scale with the number of users and resources and tolerate intermittent resource participation.

We assume that each participating site (peer) has a certain number of massively parallel machines, and each user's application is a parallel job.

2. Resource discovery and scheduling strategy

Each job request is completely described by the computing resource type description, the information about how many processors it needs and for how much time, and a deadline, that is the time period in which it has to be processed. We denote that each job j requires h_j processors for the processing time w_j. The jobs are neither preemptive nor time-sharing. Thus, once started, a job runs until completion. After its submission, a job demands a fixed number of resources for a certain time period. We do not consider the case in which a job exceeds its allocated time. This data does not change during the job execution.

A user submits jobs to the peer to which he is connected.

The scheduling component of each peer is made up of three main units: the *Request Manager*, the *Scheduler* and the *Information Provider* (Figure 1).

The *Request Manager* is the component responsible for the analysis of the received user request. This module is able to understand the structure of each received request whether it is sent by a final user or whether by an other site. It evaluates the request in order to understand if the site that it manages is capable of offering the requested services. If the site's characteristics match with the users' request, then the request is sent to the Scheduler, otherwise it is sent to the Information Provider.

The *Scheduler* is the component responsible for the mapping on the computing resources of the jobs characterized by a computing resource type description that matches with the computing resource type of the site.

The *Information Provider* is the component responsible for collecting and distributing structural and functional information about the different sites of the grid; these information regard resource composition, availability and status. The information service is a fundamental block of the scheduling component of a peer, and the quality

of its information has a strong impact on the overall grid environment performance. It collects data about a subset of other peers of the grid, it queries about non-local resource availability and status, it updates the internal data base, it replies to queries coming from other sites or forwards them to other nodes in order to implement an efficient and effective resource discovery policy.

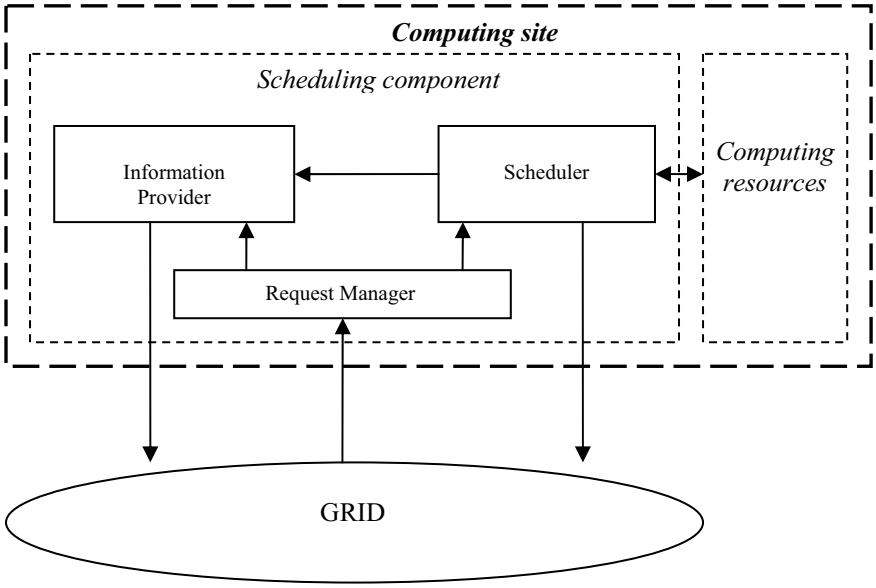


Figure 1. Scheduling component

A grid is a very dynamic and heterogeneous environment, where the role and availability of resources may rapidly change; in such environment the possibility to create a virtual working setting by collecting information about resources availability from a series of distributed and individual units might be a key aspect. This possibility is offered by an efficient information system.

The first implementations of the information provider had a centralized structure; the systems were directory-based systems, that suffered from a series of problems, including the fact that centralized servers may become bottlenecks or points of failure. Today the majority of the researchers believe that an efficient grid information service has to be based on a peer-to-peer organization.

Our work takes inspiration from the Information Service model of the Globus Toolkit, in which each unit is represented by a Grid Service, i.e. an extended Web Service following the new OGSA conventions.

Our system proposes a resource discovery and scheduling strategy characterized by high scalability and reliability.

In the following, we outline the structure of each module of our scheduling system.

2.1. Request Manager

Users submit requests for job execution to the massively computing site (peer) they are associated with. Each user request is completely defined by: a description of the specific computing resources needed (required processor's type, operating system, etc.); the number of processors; the execution time on the specific resources; and a deadline, that is the specific period in time in which the job has to be completed.

Each request is received and analysed by the Request Manager. A single request may come from a user or may be forwarded to the site from another peer. According to the different origin of the requests, the request manager has to act differently.

When a user directly submits a job to a particular peer, the Request Manager has to help the user to submit his request according to the specific format of the system. An interface is implemented, so the user can easily insert the data of his request and the Request Manager can set up a request message which has the structure needed by the system.

A message request is structured adding to the information on user constraints other specific functional information that each peer of the grid is able to easily process. These information are about: a single request's identifier, an identifier of the site that has received the request directly from the client (this site can be considered the request's owner) and a due date of the request itself.

Each request is branded with a single identifier, which avoids to analyze the same job several times. The message propagation in a peer-to-peer grid system leaves the possibility to the same request to be sent to the same peer more than once. Without this unique identifier all replications of the same request would be viewed as different jobs.

The Request Manager also adds to each request's message the identifier of the site owner of the job. The propagation of messages in a peer-to-peer grid system gives the possibility to quickly find a site capable of processing each request. To do that, it is necessary to regularly update the information about the grid environment. Each time a site is able to schedule a request, this information has to be send to the request's owner. We need to establish a mechanism to send back to each site the right information. This mechanism is based on the identifier of the site owner of each request.

The Request Manager also associates to each request a deadline for the propagation of the message. In a peer-to-peer grid organization it is necessary to establish a mechanism to stop the propagation of each message. Each time a Request Manager receives a request to which a not valid deadline is associated, the job is just skipped.

If a user directly submits a job to a specific site, the Request Manager module first sets up a request message, and then evaluates the request. This evaluation is very simple because all the requests have the same structure. Therefore, the Request Manager can understand if the site that it has to manage has the computing resources requested by the user: in the affirmative case the user's request is sent to the scheduler, otherwise it is sent to the information provider, in order to search for another site in which that computing resources may be available.

The Request Manager can also receive requests sent by other sites. In this case a request is still organized according to the standard structure described above and the Request Manager has just to evaluate it. The evaluation procedure is made up of more than one step: first the Request Manger has to check if the user request has already been analyzed or if it is a new local demand, then it has to check if the received request is still valid. The mechanism that avoids to analyze several time the same request is

based on the unique identifier that brands each request. The request manager checks if this request has been submitted for the first time to the local peer and only in this case the request is not rejected.

As pointed out above, to each request is associated also a deadline value; if it has already expired, the request cannot be processed and it is just rejected. After all these inspections/checks, the Request Manager module finally decides to which module the user's request has to be sent. Consequently, it sends it to the appropriate module: if the site managed by the Request Module has the computing resources required by the user, the request is sent to the scheduler module, otherwise it is addressed to the information provider.

2.2. Scheduler

The scheduler is the module responsible to find an appropriate strategy to map the user applications to the computing machines controlling the access to the resources and the execution of the parallel application subject to some set of optimizing criteria. Examples of scheduling criteria include maximizing the resources utilization, minimizing the expected runtime of a task set, minimizing the costs, minimizing the total communication delay, and so on.

When the Scheduler receives a user request from the Request Manager, the site has the possibility to process the submitted job. Therefore, it has to check if it is capable of satisfying the Quality of Service (QoS) required. For our application, this means that the scheduler has to be sure that it can process the job before its due date. If it is possible the job is scheduled, in the opposite case the request is sent to the Information Provider.

Since McNaughton in 1959 [18] introduced the problem of scheduling n jobs on m identical parallel processors, the problems of task scheduling on heterogeneous processors have received great attention from the research community. The great part of these studies have been concentrated on studying the scheduling problems on identical parallel machines. In 1990 Cheng and Sin [19] review the most important research results in deterministic parallel-machine scheduling theory. The most recent reviews of the research of parallel-machine scheduling have been performed by Mokotoff in 2001 [20], Gordon et al. in 2002 [21], Cheng et al. in 2004 [22].

In 1979 the task scheduling problem has been proven to be, in the most general case, NP-complete [23]. In literature there is a great number of heuristics to address the problem. These heuristics are classified into: list scheduling algorithms [24][25][26][27], duplication based algorithms [28][29], clustering algorithms [30][31], and meta-heuristics algorithms [32][33][34][35].

The basic idea of the list scheduling algorithms is to pre-process all the tasks that have to be scheduled in order to assign a priority to each one and, then, to order them into a list. Tasks are selected from this list basing on their priority and each selected task is scheduled to a suitable processor which minimizes its completion time. Some of the most famous list scheduling heuristics are: the Heterogeneous Earliest Finish Time procedure (HEFT), the Levelized Min Time procedure (LMT), the Critical-Path-On-Processor procedure (CPOP). This class of algorithms is quite simple to implement and generally they provide schedules of good quality.

The main idea behind duplication based scheduling algorithms is to utilize processor idle time to duplicate predecessor tasks and thereby reduce the waiting time of the dependent tasks.

The basic idea of the Clustering algorithms is to take into consideration the relationship between parallelism and inter-process communication management: this class of algorithms tries to schedule heavy communication tasks onto the same processor, even if other processors are available. The scheme of these heuristic is based on three steps: in the first step all the tasks that need heavy communication are grouped into a set of clusters; in the second step clusters are mapped onto the set of available processors; in the third phase cluster merging or de-clustering is done basing on the available number of processors.

In the meta-heuristics algorithms it is possible to find: Tabu Search algorithms, Simulated Annealing algorithms and Genetic Algorithms. Among this class of algorithms Genetic Algorithms seem to provide schedules of better quality.

The scheduler module works as it follows. We assume that each site (peer) of the grid is characterized by a single massively parallel machine consisting of several nodes. We also suppose that the user's requests are parallel applications that require to be allocated to some subset of processors of the parallel machine.

As expected, operations research methods become necessary in order to support advanced feature like high Quality of Service.

According to the scheduling problem theory, also the scheduling problem in a parallel resource environment has to be described in a standard way.

It is convenient to introduce the notation of Graham et al.(1979) [23] and Allahverdi et al.(1999) [36]. According with this notation each scheduling problem is denoted by the standard three-field notation $\alpha|\beta|\gamma$, where α describes the processors system, β describes the job system, γ illustrates the optimality criterion.

The first field portrays the processor set and its characteristic.

We assume that processor's set P consists of m elements: $P = \{P_1,...,P_m\}$. Generally it is possible to distinguish between parallel and dedicated processors. Each processor of a parallel system can process any job. Therefore, a job requires only a number of arbitrary processors. Also in this case it is possible to distinguish between identical parallel processors and uniform parallel processors.

Identical parallel processors are characterized by homogeneous environments, e.g. homogeneous computers and input/output devices, software licenses, personnel with same skill levels.

Uniform parallel processors are characterized by processors with different speeds, e.g. heterogeneous computers and input/output devices, network devices with different bandwidth, different skill levels within the same domain, network with different sniffing capabilities, and vulnerability analyzers with different capabilities.

The dedicated processors are usually specialized for the execution of certain functions. In this environment a multiprocessors task requires specific processors not just some number of them.

The first field of the three-field description, denoted above with the symbol α, can contain three fields: α_1, α_2, α_3. The first one describes the type of the processors; the second tells us if in the definition of the problem the number of processors is fixed or not; the last advises if the processors are always available or they are available only during particular time windows.

We suppose that each site is made up of a certain number of identical parallel processors, the number of processors is fixed and all the processors are always available.

The second field of the standard description describes the task set and its characteristics.

We assume that each job consists of n task, each task t_i $(i=1,...,n)$ can be defined by a number of parameters. While some parameters are defined just as in the classical theory (ready time r_i, due-date d_i), other parameters need to be introduced. Some examples are:

The set of simultaneously required processors fix_i, or the *family of alternative processors* set_i. A generical multiprocessor task can require simultaneously a set fix_i of dedicated processors. Sometimes it happens that a task can be executed by more than one set of alternative processors set_i.

The *number of simultaneously required processors* $size_i$, or the *maximum number of usable processors* δ_i. We have to notice that only in the case of parallel processors just one of these two parameters is given. If the first parameter is given, then task t_i can be executed only on $size_i$ simultaneously; when the second parameter is given the multiprocessors task can be executed by some number of processors in the range $[1, \delta_i]$.

Execution time. It is the time that task t_i requires to be processed. Obviously, this parameter is described in a different way in the case of dedicated processors or parallel processors. We need also to know if the tasks are preemptable or not.

The second field of the three-field description (β), can contain six fields: β_1, β_2, …, β_6. β_1 describes the type of the multiprocessor task. β_2 denotes presence or absence of preemptability and variability of the profile. β_3 explains the type of precedence constraints. β_4 informs us if processing time of task are fixed or arbitrary. β_5 lets us know if tasks have different or identical ready times. β_6 describes the type of additional resource requirements.

We suppose that each task is a static multiprocessor task characterized by the required processors' number and its execution time on these computing resources. The tasks are neither preemptable nor "malleable" in their shape. We suppose each job is composed by only one task, so there is no need to explain the type of precedence constraints because we have a group of independent tasks that have to be scheduled on a given site. We suppose also that the processing time of each task is a fixed value, and all tasks are available just when the request arrives to the site. Therefore, each request is not characterized by ready times. Our problem is characterized by an additional constraint: to each task is associated a due date.

The third field of the standard description defines the optimality criterion. It could be a standard optimality criterion, represented by the standard notation like C_{max} (Minimum Makespan), L_{max} (Minimum Max-Lateness), etc. If the optimality criterion is not standard, it has to be completely described.

In our problem the goal is to maximize the resources utilization level of the site.

To completely define the problem it is necessary to add to the three-field description another information regarding the modality of managing the data. There is the possibility for the scheduler to pre-analyze the user requests in order to better map them on the resources (off-line approach). The opposite situation is when the scheduler must make mapping decisions one request at a time (on-line approach).

In the following, we focus on the On-Line mapping resource problem. The notion of on-line scheduling [37][38][39] is intended to formalize the realistic scenario where the scheduler, unlike the off-line approaches, has not access to the whole input instance but it learns the input as soon as it becomes available. According to this model, the scheduler model does not know in advance the jobs: each job has specific requirements

for its execution, they become known to the schedulers only when the job is managed by them; the schedulers have to make decisions in real time taking into account the available suitable processing resources and cannot order the incoming job queue in any way.

We want to introduce a modular structure for the brokering and scheduling problem. According with this goal, each module can be implemented in many different ways. In literature each method proposed to solve the scheduling problem might be used to implement the "scheduler" module.

In the following, we introduce a new method to address the scheduling problem basing on a packing approach. The study of this method needs to be further investigated. The proposed algorithm has to be tested on large instances and compared with other methods already available in literature. However, at the current stage our algorithm shows to be capable of returning very good output solutions with acceptable computational times.

A brief description of our packing method for the scheduling problem together with some computational experiments conducted so far is given in the following.

We propose a processor allocation algorithm based on two-dimensional (2D) packing to solve the on-line resource mapping problem of each scheduler module.

According to this approach the mapping problem can be defined as a multiprocessor task scheduling problem where a set H of processors, and a set J of non preemptive jobs are given; the objective is to schedule without interruptions as many jobs as possible maximizing the processor total busy time. Please notice that every job is characterized by a due date value.

It is possible to model a grid computing site as a set of H identical processors available for a time window of length W [40].

If it is likely to assume that the processors are indexed and each job j requires h_j processors for the processing time w_j, then it is possible to consider the processing time of a job and the required number of processors as respectively the width and height, of a rectangle (or box).

Therefore, scheduling job j corresponds to assigning a box of height h_j and width w_j in a bounded rectangular area of height H and width W, with the objective of using such bounding area as much as possible.

More formally, the rectangle packing problem is defined as follows. Given a set I of two-dimensional rectangular-shaped boxes, where each box $j \in I$ is characterized by its width w_j and its height h_j, the problem consists in orthogonally packing a subset of the boxes, without overlapping, into a single bounding rectangular area of width W and height H, maximizing the efficiency ratio ρ between the area occupied by the boxes and the total available area A, i.e. minimizing the wasted area of A.

Two dimensional packing has been studied by many researchers as a solution to resource (such as memory) allocation problem [41][42]. But the packing problem we use in our processor allocation is somewhat different from others.

The first difference is represented by the choice of taking into account the realistic scenario in which the time goes by: our problem turns into a particular strip packing, in which the available packing area changes over time. Figure 2 shows a representation of the packing space: at each instant t, it is possible to distinguish the past from the present, and, according to this consideration, it is necessary to establish that a part of the packing area cannot be used any longer even if available.

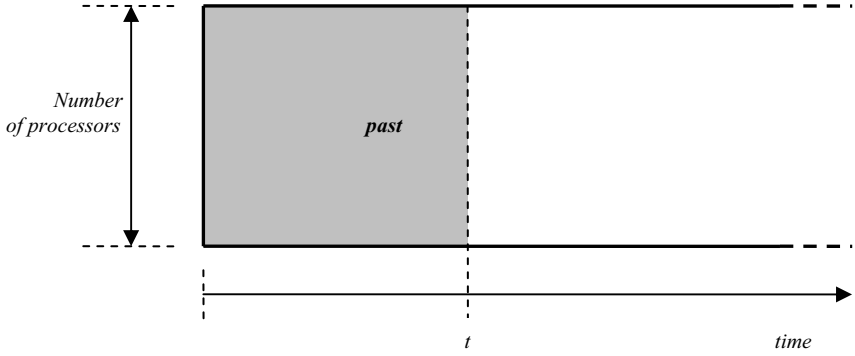


Figure 2. Description of the packing space

The second characteristic is the presence of a due date value for each job; these constraints force the scheduler to consider available for each job only a particular portion of the entire packing area.

Two different jobs are associated to different and potentially available packing area, also if their requests are analyzed in the same time period.

The third difference is represented by the choice to solve the packing problem with an on-line approach. As we have pointed out the notion of on-line approach is intended to formalize the scenario in which the jobs are not known in advance by the system. They are assumed to be ordered in a queue and presented one by one to the system according to their position in the list. In this scenario we do not have the possibility to order the items before to pack them.

Like most packing problems, our particular 2D rectangular strip packing is NP-hard. We propose a heuristic method.

To solve any particular packing problem, exact and heuristic methods have to make a crucial solution decision: where to place the items inside the bins; in other words they have to find a rule for packing items.

Having in mind the idea that the performance of a solution method is strictly dependent upon an efficient definition of the points where to place the items, we propose a heuristic algorithm based on a new rule for our particular 2D rectangular strip packing problem.

Each time a job request arrives, the scheduler acts in three phases. In the first one the feasible packing area of the candidate job is identified taking into account the time and deadline constraints. In the best case this portion of the total space is all available, but generally other jobs have been previously packed in it. At the end of the first step, the method defines the feasible packing area associated to the candidate job. In the next steps, it is necessary to establish the exact point in which the submitted job has to be placed.

Each point in which a candidate job can be placed is defined *packing point*. In the second step of the heuristic a set of candidate packing points is found. To define these points, we consider the envelope of the jobs already packed in the feasible packing area of the candidate job. A packing point is a point in which the envelope changes from vertical to horizontal. Among all found packing points we consider only the ones guaranteeing the respect of all operative constraints (deadlines, starting times, etc.). At

the end of the second step a list of candidate feasible packing points is given; if the list is empty this means that it is impossible to schedule the submitted job.

The third step of the algorithm consists in the evaluation of all the feasible packing points found and the point that has proved to be the best one is chosen. A packing point associated with a job request is a brief information of a potential starting time of its execution and the potential processors that can execute it. To each packing point is associated a different schedule. The "best" packing point associated to a job request represents the schedule which assures to best utilize the computing machines without wasting computing time. In other words, we choose the packing point that assures that scheduling the job starting from it we do not generate empty spaces (holes) into the *final envelope*. The final envelope is the new envelope obtained just adding to the previous one the new submitted job (i.e. its packing representation).

When a submitted job is scheduled, a new request can be analyzed.

The scheduler module sends a message to the request's owner site. If the request has been successfully scheduled (it means that it has been possible to respect the time constraints of the request) this success is highlighted.

2.3. Information Provider

When a user request is received by the Information Provider, it starts the resource discovery process. To do that, the module interrogates its database in order to know if there are in the grid other peers capable of providing the demanded computing services. If there are peers that potentially can provide the needed quality of services, the Information Provider forwards to them the user request, and waits for responses.

The management of these responses is an important part of the Information Provider activity, because it gives the chance to the module to increase its knowledge from the relationship with the grid environment. It is an easy way to automatically learn information regarding resource composition, availability and status. This mechanism makes the information system an intelligent component, characterized by the capability of self-learning about its environment.

As pointed out above, the Information Provider module is constructed on the basis of the efficient use of a database and a database manager module. Each database contains only a small part of the grid environment information. It is impossible to think that each computing site of a World-Wide grid, based on a peer-to-peer model, perfectly knows the actual resource composition of all the other sites located all around the world. Each node has a limited vision of the grid environment; it knows only a small set of the grid computing sites. This limited view is due not only to the huge dimension of a grid but also to its totally decentralized and changeable nature. The information stored into the database are organized in a particular way and can be updated during the lifetime of the system.

The intelligent part of the information system is the database manager module. It offers an interface to the database that allows a rapid access to the local information, but it is much more than this. It allows the self-learning mechanism, because it implements methods for information update and deletion, and according to this information it is able to recognize among all known sites the more active ones (the sites that more than the others have answered positively to the forwarded user's requests, or more than the others have decided to forward user's request).

The information stored into the database are organized by computing resources types. In particular to each computing resource type (remember a specific computing

resource type is characterized by processor's type, operating system information and so on) is associated a list of site's identifiers. Each site's identifier is branded with a priority value; the higher is this value the higher is the possibility that the associated site is able to schedule a job request in respect of the user's constraints. In the following we show how to calculate this value and how to update it during the system lifetime.

The set of potentially feasible sites for each computing resource type is divided into two parts.

In the first one are stored the sites that positively replied to similar user's requests previously forwarded to them. The higher is the number of affirmative responses received over time, the higher is the associated priority value. Each time the Information Provider receives a reply from an other peer of the grid, the data set has to be updated. In particular, this data set review varies depending if the site is already known or not. If a site is not known, its identifiers is added to the list. If the site is capable of satisfying the time constraints of the submitted request the site's identifier is added to the list with a high priority value, in the other case it receives a low priority value.

If the site's identifier is already in the list then its associated priority value has to be recalculated according to its capability of guaranteeing the respect of the time constraints of the last user requests received.

In the second part of the set are stored the sites that recently forwarded to the site the same type of user's requests. The idea is that even if they are not able to guarantee the respect of the user constraints, their databases probably have been correctly updated; so there is the possibility to find in two steps the right site to which send the request. Obviously, the information stored in this second set is much more variable than the information of the first one. The reason is that the time has a strong impact on the information of this set; time goes by so this set has to be frequently updated: new forward request has to be added into the list and the priority of the oldest one has to be decreased.

According to the classification defined by the database manager each time a user's request have to be forwarded, the Information Provider is able to identify a set of sites that potentially are capable of guaranteeing the desired Quality of Service and/or a set of sites that can give updated information about the environment. The number of sites in these sets is a dynamic variable that must be decided each time taking care of the deadline value of the request; more strict is this value greater is the number of sites to which the Information Provider forwards the request.

According to this information analysis, the request's message is forwarded to the identified sites of the grid.

3. Evaluation

The current work is at an evolutionary stage: relative efficiency is under evaluation with the aim to assess the merits of the proposed approach. Only the efficiency and effectiveness of the proposed processor allocation method of each computing site has been evaluated. The results demonstrate that the used packing approach can achieve high objectives in terms of user application satisfaction degree and computing resources utilization level. Further analysis is necessary to validate the presented model

in order to incorporate resource discovery method and show the appropriateness of the overall solution methodology.

We propose a processor allocation algorithm based on a particular two-dimensional strip packing to solve the on-line resource mapping problem of each scheduler module. The particular nature of this strip packing problem does not allow the comparison of our solution method to the conventional packing algorithms.

The computational tests, provided on packing representation of computational grids and of a set of parallel user applications, have proved that the heuristic approach always finds out a feasible schedule if one exists. It is interesting to notice that in the on-line approach the existence of a feasible solution depends on the particular relationship between the resource availability of the computing site and the deadline constraint of each user application. Since it is an on-line approach it is not possible to order the submitted jobs according to any rule with the goal to maximize the resource exploitation. Therefore, each time a job request arrives the scheduler module schedules it taking into account the resources available at that moment and the schedule decisions already made. Considering the resources availability, it may happen that the deadline of a submitted job cannot be respected, in these cases the heuristic approach has to just respond that it is impossible to schedule the particular application. Otherwise, if the particular time constraints of the submitted job make possible to find a feasible solution, then the proposed method looks for the best possible schedule at that moment.

For our simulations, the processing time and the number of processors requested by each job were generated using the Poisson distribution. The deadline values have been calculated taking into account the processing times, in particular each deadline value of a job is a multiple of a processing time of the job [43][44].

For our simulations, as functional characteristics of each computing site we took the industry values selected on currently commercially available parallel computing machines [45].

We measured the site utilization degree values obtained by applying different job workflows. These values indicate how tightly the sites are packed. In a given time period, the processor utilization is the ratio of the number of processors used in allocation to the total number of processors or, if you prefer, the ratio of the sum of the area of all the scheduled jobs to the total available area of the site.

We repeated the experiment 100 times. Figure 3 shows the average site utilization degree.

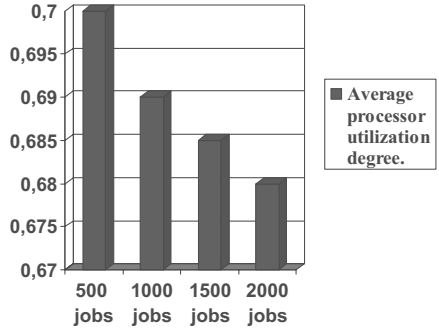


Figure 3. Processor utilization degree

We measured also the user application satisfaction, which indicates the portion of the user requests that the system is able to schedule. In a given time period, user application satisfaction is the ratio of the number of scheduled user application to the total number of submitted request. Figure 4 shows the user application satisfaction degree values obtained by applying different job workflows. We repeated this experiment 100 times.

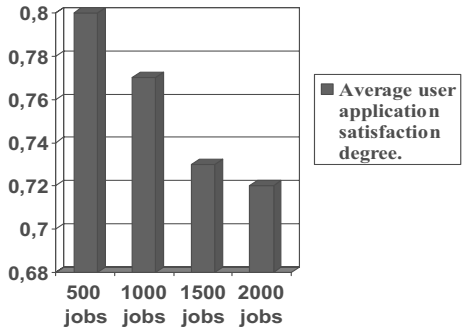


Figure 4. User application satisfaction degree.

4. Conclusion

In the future the grid will be a vast, heterogeneous computing environments supposed to provide computational services on demand to users located all over the world without requiring any other information. The technological progress during the last decades let us think that this future is not so far. In this dynamic environment, information about the configuration and status of the distributed resources is extremely variable and consequently precious in order to achieve an overall efficiency.

It is necessary to design a versatile resource discovery and scheduling system able to satisfy a potentially very large number of users tolerating delays and faults and updating very quickly.

We have proposed a new fully decentralized solution for both resource brokering and scheduling in a grid environment based on a peer-to-peer model. We have considered these two steps as a single activity in order to achieve both short response times and high percentages of computing resources utilization.

The proposed resource discovery solution uses the relationship of each node with its neighbours to learn structural and functional information about the resource-sharing environment. These data connect information regarding the configuration and status of the distributed resources; the status information are derived from the relationship between neighbours, in particular taking into account all the scheduling solutions of each single computing site. These data, stored in appropriate data bases, are queried by database managers in order to know which peers in the grid are capable of providing the demanded computing services supporting advanced features such as high Quality of Service.

Further investigation is necessary in order to show the appropriateness of the overall solution methodology. A logical and functional validation of the solution

methodology shows that the system scaled effectively with the dimension of the computing environment and leaves to the user a high flexibility in defining the desired computing services. Since the modular structure of our approach, problems like site autonomy, heterogeneous computing services and individual provider policies can be easily addressed. Additionally, features as co-allocation and multi-site scheduling over different resource domains can be supported.

Future work will incorporate more extensive analysis of the entire system.

References

[1] Foster, I., Kesselman, C., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999.

[2] Foster, I., What is the Grid? A Three point checklist**,** 2002. http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf.

[3] Foster, I., Kesselman, C., Nick, J., Tuecke, S., The physiology of the Grid: An open Grid services architecture for distributed systems integration, 2002. http://www.globus.org/research/papers.html#OGSA.

[4] Foster, I., Kesselman, C., Tuecke, S., The Anatomy of the Grid: Enabling scalable virtual organizations, *International Journal of Supercomputer Applications* **15(3)** (2001), 200-222.

[5] Foster, I., Internet Computing and the Emerging Grid, *Nature Web Matters* 2000. http://www.nature.com/nature/webmatters/grid/grid.html.

[6] Gradwell P., Overview of Grid Scheduling Systems. http://peter.gradwell.com/phd/writings/computing-economy-review.pdf

[7] Lim B., Wen J., Web Services: An Analysis of the Technology, its Benefits, and Implementation Difficulties, *Information Systems Management* **20 (2)** (2003), 49-57.

[8] Foster, I., Kesselman, C., Globus: A metacomputing infrastructure toolkit, *International Journal of Supercomputer Applications* **11(2)** (1997), 115-128.

[9] I. Foster, and C. Kesselman, The Globus Project: A Status Report, in *Proceedings of Heterogeneous Computing Workshop*, IEEE Press (1998), 4-18.

[10] The Globus Alliance: Globus toolkit 3, globus information services documentation, 2004. http://www.globus.org/mds/.

[11] Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S., Condor-G: A Computation Management Agent for Multi-Institutional Grids, *Cluster Computing*, volume **5 (3)** (2002), 237-246.

[12] Litzkow, M.J., Livny, M., Condor - a hunter of idle workstations, in *Proceedings of the 8th IEEE International Conference Distributed Computing Systems* (1988), 104–111.

[13] Clark, D., Face-to-Face with Peer-to-Peer Networking, *Computer* **34 (1)** (2001), 18-21.

[14] Schopf, J.M., Grid Resource Management - State of the Art and Future Trends, in *Actions When Grid Scheduling - The User as a Grid Scheduler*, Kluwer Academic Publishers (2003), 15–23.

[15] Krauter, K., Buyya, R., Maheswaran, M., A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing, *International Journal of Software: Practice and Experience (SPE)* **32(2)** (2002), 135-164.

[16] Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C., Grid Information Services for Distributed Resource Sharing, in *Proceedings of 10th IEEE Symposium on High Performance Distributed Computing* (2001), 181-194.

[17] Czajkowski, K., Foster, I., Kesselman, C, Martin, S. Smith, W., Tuecke, S., A resource management architecture for metacomputing systems, *Job Scheduling Strategies for Parallel Processing*, *Lecture Notes in Computer Science* **1459** (1998), 62–68.

[18] McNaughton, R., Scheduling with deadlines and loss functions, *Management Science* **6(1)** (1959), 1-12.

[19] Cheng, T.C.E, Sin, C.C.S., A state-of-the-art review of parallel-machine scheduling research, *European Journal of Operational Research* **47(33)** (1990), 271-292.

[20] Mokotoff, E., Parallel machines scheduling problems: A survey, *Asia-Pacific Journal of Operational Research* **18** (2001), 193–242.

[21] Gordon, V., Proth, J.M., Chu, C., A survey of the state-of-the-art of common due date assignment and scheduling research, *European Journal of Operational Research* **139(1)** (2002), 1-25.

[22] Cheng, T.C.E., Ding, Q., Lin, B.M.T., A concise survey of scheduling with time-dependent processing times, *European Journal of Operational Research* **152(1)** (2004), 1-13.

[23] Graham, R.L., Lawler, L.E., Lenstra, J.K., Kan, A.H., Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, *Annals of Discrete Mathematics* **5** (1979), 287-326.
[24] EI-Rewini, H., Lewis, T.G., Scheduling Parallel Program Tasks onto Arbitrary Target Machines, *Journal of parallel and Distributed Computing* **9** (1990), 138-153.
[25] Iverson, M., Ozguner, F., Follen, G., Parallelizing Existing Applications in a Distributed Heterogeneous Environments, *Proc. Heterogeneous Computing Workshop* (1995), 93-100.
[26] Kwok, Y., Ahmed, I., Dynamic Critical-Path Scheduling: An Effective technique for Allocating task graphs to Multiprocessors, *IEEE Trans. on Parallel and Distributed Systems* **7(5)** (1996), 506-521.
[27] Topcuglou, H., Hariri, S., and Wu, M.Y., Performance Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, *IEEE Trans. on Parallel and Distributed Systems* **13(3)** (2002), 263-274.
[28] Ahmed, I., Kwok, Y., On Exploiting Task Duplication in Parallel Program Scheduling, *IEEE Trans. on Parallel and Distributed Systems* **9(9)** (1998), 872-892.
[29] Rashmi Bajaj, Agrawal, D.P., Improving Scheduling of Tasks in a Heterogeneous Environments, *IEEE Trans. on Parallel and Distributed Systems* **15(2)** (2004), 107-118.
[30] Gerasoulis, A., Yang, T., A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs onto Multiprocessors, *Journal of Parallel and Distributed Computing* **16(4)** (1992), 276-291.
[31] Kafil, M., Ahmed, I., Optimal Task Assignment in Heterogeneous Distributed Computing Systems, *IEEE Concurrency* **6(3)** (1998), 42-51.
[32] Ahmad, I., Dhodhi, M.K., Multiprocessor Scheduling in a Genetic Paradigm, *Parallel Computing* **22** (1996), 395-406.
[33] Hou, E.S., Ansari, N., Ren, H., A Genetic Algorithm for Multiprocessor Scheduling, *IEEE Trans. Parallel and Distributed Systems* **5(2)** (1994), 113-120.
[34] Dhodhi, M.K., Ahmad, I., Yatama, A., An Integrated Technique for Task Matching and Scheduling onto Distributed Heterogeneous Computing Systems, *Journal of parallel and distributed computing* **62** (2002), 1338-1361.
[35] Tsuchiya, T., Osada, T., Kikuno, T., Genetic-Based Multiprocessor Scheduling Using Task Duplication, *Microprocessors and Microsystems* **22** (1998), 197-207.
[36] Allahverdi, A., Gupta, J.N.D., Aldowaisan, T., A review of scheduling research involving setup considerations, *Omega* **27(2)** (1999), 219-239.
[37] Shmoys, D., Wien, J., Williamson, D.P., Scheduling parallel machines on-line, *SIAM Journal of Computing* **24(6)** (1995), 1313-1331.
[38] Fiat, A., Woeginger, G.J., Online algorithms: the state of the art, *Lecture Notes in Computer Science* **1442**, 1998.
[39] Borodin, A., El-Yaniv, R., Online computation and competitive analysis, Cambridge University Press, 1998.
[40] Caramia, S. Giordani, A. Iovanella, Grid scheduling by on-line rectangle packing, *Networks* **44(2)**, (2004), 106-119.
[41] Baker, B. S., Brown, D. J., Katseff, H. P., A 5/4 Algorithm for Two-Dimensional Packing, *Journal* of *Algorithms* **2** (1981), 348-368.
[42] Baker, B. S., Schwarz, J. S., Shelf Algorithms for Two-Dimensional Packing Problems, *SIAM Journal on Computing,* **12 (3)** (1983), 508-525.
[43] Takefusa, A., Casanova, H., Matsuoka, S., Berman, F., A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid, in *Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing* (2001), 406-415.
[44] Caron, E., Chouhan, P.K., Desprez, F., Deadline Scheduling with Priority for Client-Server Systems on the Grid, in *Proceedings of 5th IEEE/ACM International Workshop on Grid Computing* (2004), 410-414.
[45] Dongarra, J.J., Performance of Various Computers Using Standard Linear Equations Software, Technical Report University of Tennesse CS-89-85 (2007).
http://www.netlib.org/benchmark/performance.pdf

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Challenges of Scale: When All Computing Becomes Grid Computing

Robert J. FOWLER, Todd GAMBLIN, Gopi KANDASWAMY, Anirban MANDAL, Allan K. PORTERFIELD, Lavanya RAMAKRISHNAN and Daniel A. REED
{rjf, tgamblin, gopi, anirban, akp, lavanya, reed}@renci.org

Renaissance Computing Institute (RENCI)
University of North Carolina
Chapel Hill, North Carolina 27517 USA

Abstract. Scientific computing is moving rapidly from a world of "reliable, secure parallel systems" to a world of distributed software, virtual organizations and high-performance, though unreliable parallel systems with few guarantees of availability and quality of service. This transformation poses daunting scaling and reliability challenges and necessitates new approaches to collaboration, software development, performance measurement, system reliability and coordination. This paper describes Renaissance approaches to solving some of today's most challenging scientific and societal problems using Grids and parallel systems, supported by rich tools for performance analysis, reliability assessment and workflow management.

Keywords. Petascale, mesoscale weather, bioinformatics, performance analysis, reliability, workflows, multidisciplinary science

Introduction

A brave new world of computing is emerging, driven by conflicting technical and economic forces. As many have observed, the strict form of Moore's law (i.e., a regular doubling of transistor density on chips) continues apace. However, this is no longer synonymous with rising clock rates, as chip power consumption is now a critical limiter. More perniciously, after deep pipelining and superscalar instruction issue, there is little additional instruction level parallelism (ILP) available to further increase single thread performance.

The consequence of these technology trends is an increasing emphasis on other forms of on-chip parallelism, notably multi-core designs. Intel and AMD already offer quad-core designs [1]; chips with 80 or more cores are expected within a decade. Equally importantly, these designs are being augmented with multi-threading to increase memory latency tolerance for deep memory hierarchies. To achieve high

application performance on such chips, one must exploit this parallelism aggressively at multiple levels.

Complementing chip technology evolution is the ongoing refactoring of software processes as Grid and web services, driven by low cost broadband networks and standard services, as well as the increasing need to solve multi-disciplinary problems involving geographically distributed data. This rise in software complexity, with concomitant integration, optimization and reliability problems, exacerbates the challenges of low-level multi-core parallelism.

From ubiquitous web mashups to latency tolerant distributed scientific applications (e.g., SETI@home [2] and Folding@home [3]) that can be decomposed into many small but independent computations with limited input and output data, standard web service APIs enable rapid assembly of complex software systems. Solving another class of high-end science and engineering problems entails more complex coordination – orchestrating many long-running, dependent computations that both consume and generate large data sets [4]. These ensembles or workflows (e.g., mesoscale weather modeling [5]) require stateful Grid services [6] and larger-scale computing infrastructure. Other cases include coupled, multi-scale simulations and engineering optimization problems, where each evaluation of the objective function is itself an expensive, parallel simulation.

All of these distributed computations are driven by the explosive growth of scientific data from a new generation of instruments and the rise of national and multinational teams developing data analysis and computational workflows that span organization boundaries. Such geographic dispersion, together with massively parallel computing systems, poses daunting scaling and reliability challenges and will necessitate new approaches to collaboration, software development, performance measurement, system reliability and coordination.

A major thrust of the Renaissance Computing Institute (RENCI) [7] is to harness the power of Grids, web services and parallel systems to solve some of today's most challenging scientific and societal problems. In this model, multidisciplinary teams of computer scientists and domain researchers work collaboratively to develop, test and apply computing technology.

With this backdrop, the remainder of this chapter is organized as follows. In §1, we begin with a motivating example from mesoscale weather forecasting, followed in §2 by a review of the hardware and software challenges inherent in building and executing such complex applications. In §3, we discuss scalable performance and reliability measurement for parallel systems and Grids. This is followed in §4 by an analysis of two complex workflows and their implications for qualitative resource specification. We conclude in §5 with a review and insights for the future.

1. Mesoscale Weather Modeling: A Motivating Example

As an example of workflow complexity, consider mesoscale weather prediction, as embodied in the Linked Environments for Atmospheric Discovery (LEAD) [8] project. Dynamically adaptive, real-time forecasting involves identifying, accessing, preparing and integrating disparate and high volumes of meteorological data streams, managing the logistical complexity of running numerous numerical models on a distributed set of heterogeneous resources, and analyzing and visualizing their output, which may in turn control on-line instruments like Doppler radars.

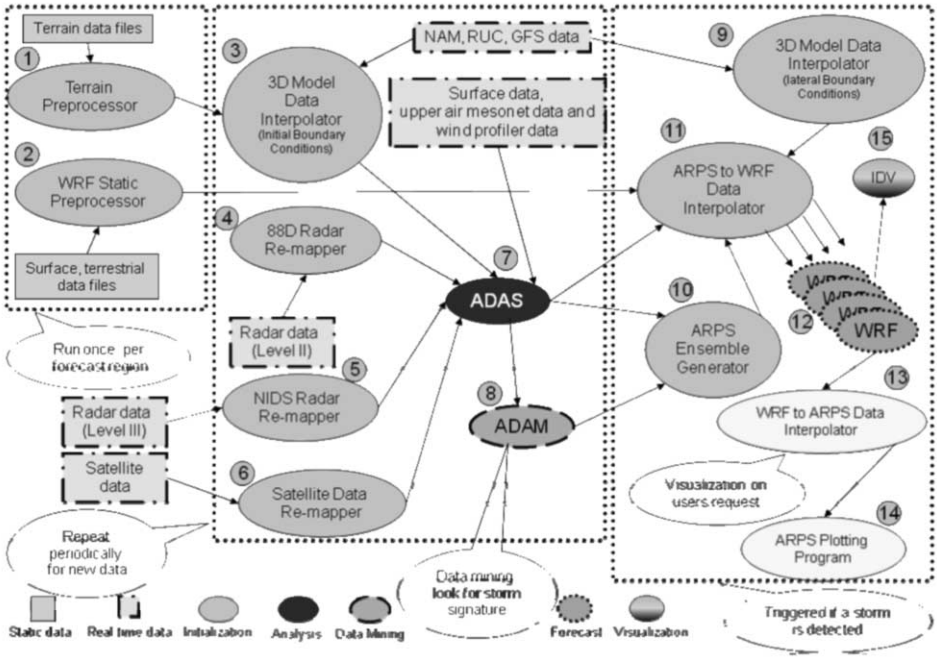


Figure 1 Dynamic Mesoscale Weather Forecasting Workflow

Figure 1 shows the phases of a typical dynamic mesoscale weather forecasting workflow: data pre-processing, data analysis, forecasting and visualization. The data pre-processing and analysis phases involve assimilating and processing large volumes of data from distributed sources (e.g., local observations, Doppler radars, hot air balloons, wind profilers, satellites and surveillance aircraft). The forecasting phase involves running large ensembles of forecast models like WRF [5] for storm, mesoscale and synoptic weather predictions.

Dynamic weather forecasting workflows primarily respond to changing weather patterns. For example, a radar network using distributed and collaborative adaptive sensing may scan potential tornado regions in a super-cell storm. In turn, this may trigger a data mining agent for detecting severe weather patterns, which might launch a workflow ensemble for tornado prediction.

In addition to weather response, dynamic workflows must also respond to the changing nature of the computational grid for 'faster than real-time' prediction of severe weather events. Changes in resource availability and services due to service or resource failures may affect the availability of certain types of data, prevent the refinement of computational meshes and reduce the number of possible computations in an ensemble. The result could be reduced forecast accuracy and timeliness.

In addition to Grid workflow development tools, realizing this adaptive forecasting capability requires a real-time distributed performance monitoring infrastructure to detect failures of applications, services and resources, and to capture performance and reliability metrics at multiple levels. This must be coupled with fault tolerance and recovery techniques and dynamic resource allocation strategies to meet soft real-time weather forecasting constraints. These tools and mechanisms are the subject of further discussion below.

2. Performance and Reliability Challenges

Hybrid workflows such as those in LEAD, which combine large parallel computations (e.g., WRF [5]), distributed data management and transfer, soft real-time constraints, security and reliability across organizational boundaries, necessitate new approaches to performance optimization, software resilience, and effective software development and abstraction. Below, we consider both topics and RENCI's multidisciplinary approach.

2.1. Resilience and Scale

The high performance and low cost of commodity components have made it practical to construct very large computing clusters containing tens to hundreds of thousands of processors. Indeed, systems with a peak performance of one petaflop, based on quad-core chips, are being built now, and national competitions are underway [9] for systems with a peak performance of ten or more petaflops containing hundreds of thousands of cores and petabytes of DRAM. Such systems will also contain tens of thousands of disks, both for aggregate storage capacity and I/O bandwidth.

Figure 2 shows the number of processors (cores) in each of the top 20 systems in the Top 500 list of supercomputers [10]. Plateaus in the "Max" curve represent evolutionary improvements in one architectural class. Downturns in the "Min" curve represent more powerful processors entering this part of the list. The processor architectures represented near the top of the current Top 500 list exhibit sustained double precision floating point performance of multiple gigaflops.

If we assume continued technology improvements, the sustained performance of high end, general-purpose cores will soon be roughly 10 GFLOPS and, a system capable of sustaining one petaflop on the Linpack benchmark will require at least 100,000 cores. As Figure 2 suggests, a system capable of sustaining a petaflop on a full application only one quarter as efficient as Linpack will require at least 400,000 cores.

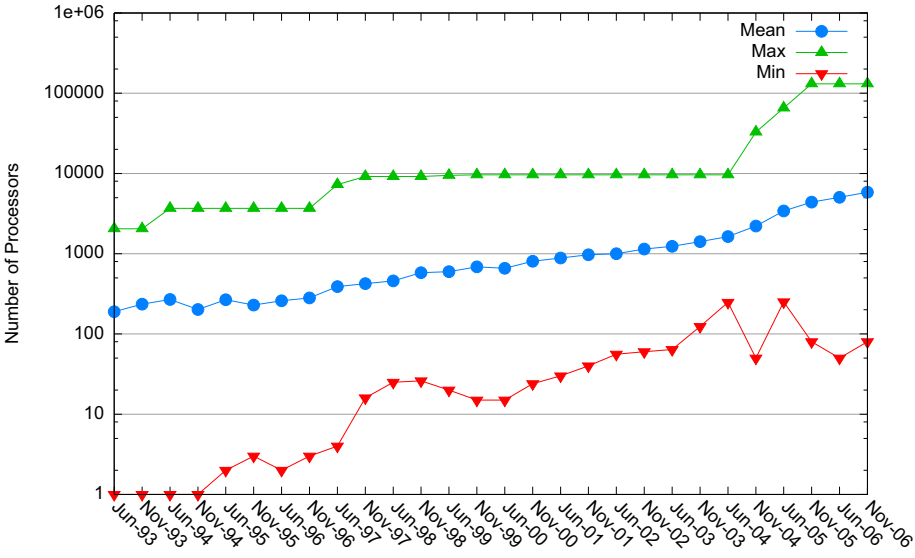


Figure 2 Top 500 Growth in System Size

Alternative designs such as the IBM's BG/L and its successors [11] that use lower performance, lower power cores will require a larger number of cores, but they can be packed more densely. Hence, all systems designed for sustained petaflop performance will contain hundreds of thousands of cores and hundreds of racks.

Increasing scale brings concomitant reliability problems [12]. Today, parallel applications either rely on standard checkpoint/restart mechanisms for failure recovery and reliability. Most workflows terminate if an intermediate step does not complete due to network failure. Likewise, most operating systems crash and reboot if a major portion of the underlying hardware fails.

As the number of failure points climbs exponentially, standard approaches are no longer acceptable. Even under generous assumptions of independent failures, the mean time to failure of any component in the aggregate can be hours or even minutes. This means that long-running parallel elements of complex workflows are unlikely to complete when executed at scale. Failure will be the norm, rather than the exception. Because stopping the system for repair will become impractical, systems and parallel applications must be designed to tolerate component failures. They need to "run past failure" and do so repeatedly.

To see why this is so, consider Google's analysis of disk failures, which showed that between 1.7 percent and 8.6 percent [13] of the disks failed each year (depending on age). For systems with ten thousand disks, a disk will fail every other day, rising to twice a day for slightly older systems. At these failure rates, not only is hardware "hot-swapping" a requirement, but the software must be able to initialize and incorporate the new hardware dynamically.

In a complementary study, we analyzed application sensitivity to transient memory errors [14], an increasingly common event for high density DRAMs with fewer electrons representing a bit in each transistor. This study showed that these errors frequently lead to application failures or more troubling, change computation outputs.

Similarly, recent analyses of Grid workflow executions [15] have shown that a non-trivial fraction of workflows fail due to improperly configured or unreliable services, transient data transfer problems or remote system unavailability. When convolved with parallel system failure modes, this strongly suggests that end-to-end fault tolerance mechanism are required.[1]

Fault tolerance mechanisms have been added to Grid workflows. Within an individual parallel application, FT-MPI [16] provides a framework for the application developer to recover from faults (communication or processor) and continue execution at a relatively low execution cost. Traditional application fault tolerance takes checkpoints of the entire program, either at the application's or the system's direction, and restarts execution [17] from the last good checkpoint after failure. Restarting from a checkpoint may cost the application minutes in lost computation. In LEAD, where a single computation of WRF takes hours, a fault tolerant version can prevent hours of lost computation, avoiding the restart of a multi-hour run.

Collectively, these studies and trends suggest that scientific computing is moving rapidly from a world of "reliable, secure parallel systems" to a world of distributed software, virtual organizations and high-performance, though unreliable parallel systems with security threats and few guarantees of availability and quality of service. For large, complex workflows, reliability and recovery are critical features that must be

[1] We will return to this topic in §3.4, where we discuss our VGrADS resilient workflow infrastructure.

built into every portion of the application, application framework, libraries and services and operating systems, and into the hardware.

The historical distinction between parallel and distributed systems will disappear with resilient non-stop services *de rigeur* and complex workflows executing atop heterogeneous collections of distributed and parallel systems. Simply put, we can expect more instances of Grid-in-a-room, Grid-in-a-box and even Grid-on-a-chip.

2.2. Heterogeneity and Complexity

The scale and diversity of complex workflows executing atop heterogeneous collections of distributed and parallel systems pose daunting challenges for every level of the software stack, from operating systems through middleware, programming environments and application frameworks. Nowhere is this more apparent than in the complexity of application development, which currently requires users to navigate a maze of workflow specifications (e.g., BPEL [18]), resource discovery (e.g., GRAM [19] and MDS [20]), scheduling (e.g., OpenPBS [21] and [22]), security and authentication (e.g., Grid certificates [23]), file transfer (e.g., GridFTP [24]) and organizational management issues.

Such complexity challenges our intuition and design capabilities. If parallel programming using MPI-style message passing is difficult, then Grid application development is daunting, particularly when coupling codes written in multiple programming languages and styles and created by multiple groups. The gulf between the simple and intuitive abstract model of a sequential von Neumann-style processor and the complexity of today's Grid workflow implementations atop clusters of multi-core processors is immense and widening.

Only by hiding this complexity and simplifying development processes can we engage a broad community of developers and scientists. New, more intuitive programming paradigms are needed that can easily and quickly express component dependencies and exploit available parallelism. Two complementary but integrated approaches are likely to address this problem: graphical interfaces for workflow specification and configuration and virtual Grid abstractions that hide the details of resource specification and selection. This approach simplifies workflow configuration, creation and execution by separating functions and hiding unnecessary complexity.

Our collaborators in the Linked Environments for Atmospheric Discovery (LEAD) project [8, 25] have developed a graphical workflow composer based on generalized BPEL that allows weather researchers, practicing meteorologists and students to configure and execute workflows without knowledge of Grid certificates, file transfers or other Grid software. They need only specify weather related parameters (e.g., target region and resolution). Figure 7 illustrates one version of this capability.

Concurrently, in the VGrADS project [26], we are developing a virtual Grid abstraction that enables workflow developers to specify desired resource performance and reliability with intuitive, qualitative descriptions. This allows resource discovery and execution systems to deliver services (i.e., performance or reliability guarantees) without identification of specific resources. Conceptually, one can request a reliable, high performance cluster rather than requesting that a specific resource be reliable. We will describe this approach in more detail in §4.3.

2.3. A Renaissance Approach

Multi-core processors, petascale systems, Grid and web services, performance tuning, reliability and fault tolerance and distributed data management – daunting challenges for scientific researchers in individual domains, but often insurmountable when researchers across multiple disciplines must collaborate to solve societal problems. From computational modeling and biomedical data analysis to coupled ocean, atmosphere and environmental models for disaster response, computational science via Grids and high-performance computing are critical enablers, but only if the tools are robust and easy to use.

By bringing together scientists, technology experts, educators, artists, humanists and business and government leaders, the Renaissance Computing Institute (RENCI) is helping address some of today's most challenging multidisciplinary problems. Key to this approach is developing not only technologies but also mutually rewarding collaborative processes for computational discovery. Computational science writ large is now the third pillar of scientific discovery [27], but it will be successful only if we can develop effective tools for software optimization and reliability and create robust software environments for creative expression. This is the RENCI vision.

3. Large-Scale Performance Analysis

Optimizing the performance and ensuring the reliability of large, complex workflows requires analysis of workflow middleware and scheduling, networking and data transfer overheads, and sequential and parallel application behavior at multiple levels. The constituent applications within workflows will be written in many languages and using many programming models. Even the workflows themselves will embody diverse programming models, from master-worker to iterative fork-join ensembles.

We are developing a suite of tools for performance and reliability assessment that targets both individual systems and applications and complex, distributed workflows. For individual systems and their parallel applications, low overhead performance and reliability measurement and intuitive bottleneck identification are critical to petascale optimization and fault tolerance. For complex workflows, distributed monitoring and fault tolerance (i.e., retry, migration or over-provisioning) increases the probability that workflows execute successfully.

Below, we describe our AMPL toolkit for statistical performance sampling of petascale applications, the HPCtoolkit for source code performance analysis, the HAPI toolkit for failure monitoring and the LEAD/VGrADS system for resilient workflow execution. Collectively, these tools enable end-to-end performance analysis and reliability assurance for petascale workflows.

3.1. Scalable Performance Measurement

Because emerging petascale systems contain tens to hundreds of thousands of processors, and the applications they host will execute for days or weeks, overly detailed performance measurement can induce unacceptable overhead and produce prodigious amounts of data. For example, a detailed trace of an SPMD application on a 1,000 processor system can reach hundreds of gigabytes in size in minutes. Such

excessive volumes stretch our capacity to store and understand performance data and manually processing such data is both tedious and time-consuming.

Traditional performance measurement techniques span a spectrum from statistical sampling to detailed event tracing. Each strikes a different balance between measurement overhead and insight into temporal dynamics. Profiling via statistical sampling is rarely adequate to identify complex performance problems and detailed trace collection on petascale systems is difficult, as I/O backbones on most petascale architectures lack the bandwidth to support concurrent output from all processors without excessive system perturbation.

To address these problems, we are exploring event tracing techniques based on population sampling [28] that retain the benefits of detailed tracing but with lower overhead and smaller data volume. RENCI's Adaptive Monitoring and Profiling Library (AMPL) [29] chooses representative processes for detailed tracing by exploiting the statistical properties of large populations. Given a population, sampling theory estimates the mean using only a small sample of the total population. For a simple random sample, the minimum sample size n that estimates the population mean with confidence z_α and error bound d is

$$n \leq N\left[1 + N\left(\frac{d}{z_\alpha S}\right)^2\right]^{-1}$$

where N is the population size and S is the sample variance.

Population sampling has long been used for social science surveys, and it has the desirable property that the number of processes one must monitor to obtain an accurate performance estimate scales sub-linearly with the total number of processes. For very large systems, only a comparatively small percentage of all processes must be sampled to get reasonable results.

As the equation suggests, the sample size also depends on the variance of monitored data. When the variance is low, fewer processes must be traced for accurate results. AMPL adaptively estimates the variance of performance data across processes and adjusts the sample size as execution progresses. Performance variation across application processes can be due to many factors, some intrinsic (e.g., data dependent behavior) and some extrinsic (e.g., network contention or operating system jitter [30]).

Fortunately, for most parallel and distributed programs, the application processes form a small number of equivalence classes (e.g., the master and the group of workers in a master-worker Grid execution model or processor zero and the others in an SPMD parallel code). When the variability is too large, however, the required sample size may grow too large for efficient data extraction. In these cases, AMPL can stratify the samples (i.e., partition the population into equivalence classes and sample each independently). Because the variance is low within each equivalence class, the number of samples (processors) required to characterize each equivalence class is also low, hence decreasing the total number needed to characterize the application.

As an example of the power of AMPL, we analyzed the performance of ADCIRC [31, 32], a FORTRAN shallow-water circulation MPI code used to model storm surges. Figure 3 shows the data volume for MPI traces from runs on a 2048-node system with varying confidence and error constraints. For 90% confidence and 8% accuracy,

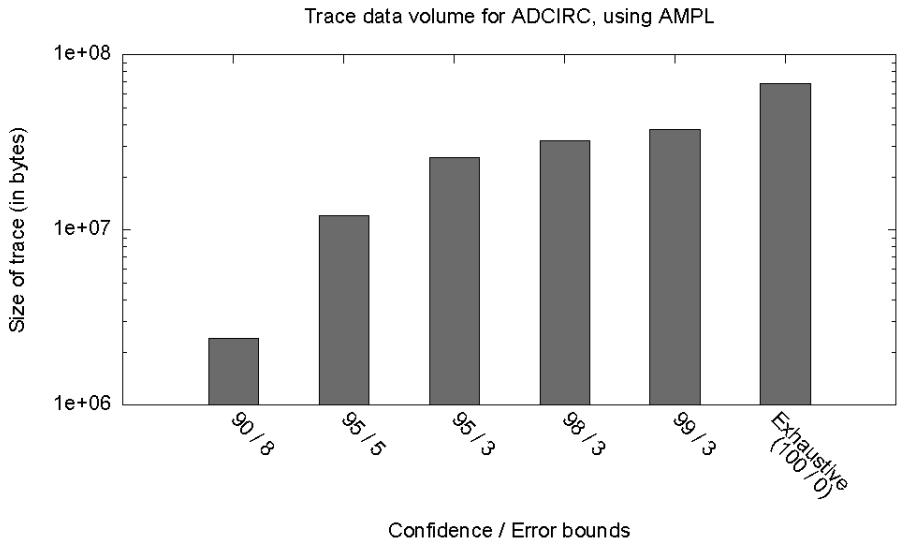


Figure 3 AMPL Performance Data Reduction

AMPL reduces total trace volume by over 28X relative to exhaustive monitoring. As confidence and error bounds are tightened, trace volume increases. The performance engineer is now free to manage this tradeoff, depending on the trace size he or she can afford. For most petascale applications, stratified sampling should reduce the overhead and trace size to the point where performance tuning is possible for production executions.

3.2. Scalable Performance Analysis

Scalable performance data capture is necessary but not sufficient. One must also extract insights from the performance data and identify both the proximate and root causes of poor performance. The simplicity of this statement belies the complexity of its realization.

The number of performance tool prototypes continues to grow, but few of these tools are used and even fewer are capable of analyzing performance data from large-scale, production applications on terascale and petascale systems. There are many reasons for this: parallel applications and systems are complex and evolving rapidly, performance problems are sometimes illusive, tools are often non-intuitive and difficult to use, and a new toolset often must be learned for each system.

The Rice/RENCI HPCtoolkit [33, 34], shown in Figure 4, is one notable exception. Designed and tested in collaboration with researchers and code developers at Los Alamos National Laboratory (LANL), the HPCtoolkit supports the key attributes of production performance analysis tools: (a) relating detailed performance data to application code without modification to either the source code or application build environments and (b) using a system-independent browser interface for generality and simplicity.

Concretely, the HPCtoolkit relies on binary instrumentation to capture performance data, correlated with standard symbol table entries (already used for

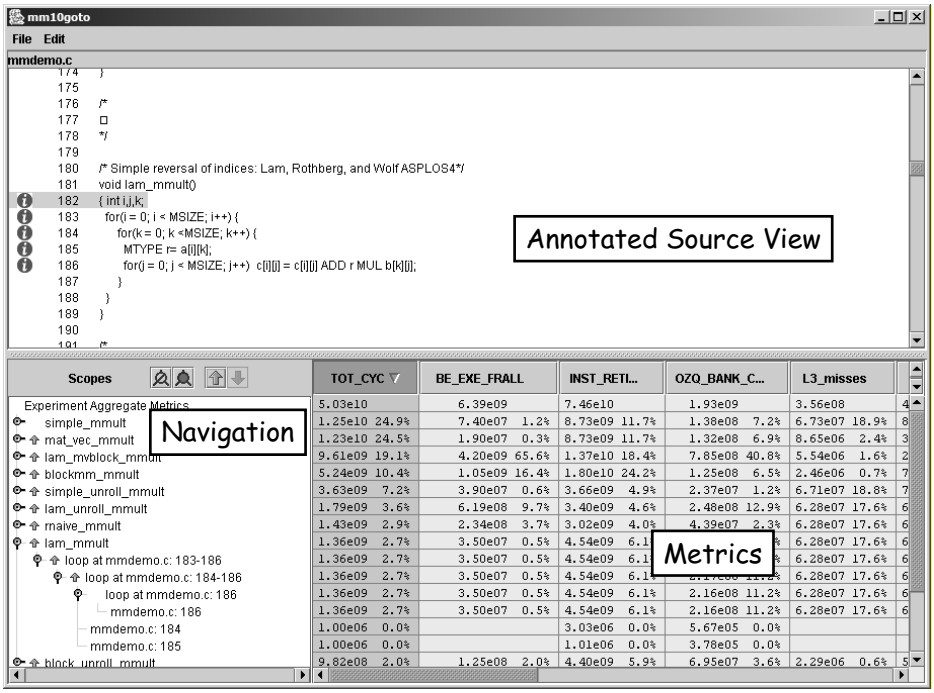


Figure 4 HPCtoolkit

debugging). No recompilation or source code modifications are required, and codes can be analyzed with all compiler optimizations enabled. Because compiler code motion (e.g., loop unrolling, software pipelining and procedure inlining) can obscure source code structure, the HPCtoolkit recreates program loop and call structures via in-depth analysis and data correlation. It also includes a statistical event counter that exploits hardware performance counters, either native counters or those from the PAPI toolkit [35], to identify loop bottlenecks and low overhead call stack tracing (three percent overhead) to identify procedure execution time distributions.

Together, AMPL and the HPCtoolkit reduce measurement and analysis overheads to a small percentage of execution time, making it practical to analyze application performance at scale. This greatly increases the utility and applicability of performance tuning. Once workflow components execute efficiently, the tuned applications become the building blocks of petascale ensemble applications.

3.3. Reliability Measurement and Assessment

As we noted in §2.1, large systems are susceptible to component failures, making continued system operation dependent on rapid failure detection and recovery. Current computer systems contain a diverse set of diagnostic monitors, sensing temperatures, electrical loads, fan speeds, memory and processor errors and disk behavior. However, these monitors have system-specific, idiosyncratic interfaces, making coordinated data analysis difficult. Similar problems existed for hardware performance counters until the PAPI toolkit [35] provided standard software interfaces for all common processor architectures.

RENCI's Health Application Programming Interface (HAPI) builds on ideas from PAPI and provides a common interface to multiple hardware diagnostic monitors. This allows higher level tools to monitor and modify environmental factors that reduce reliability. For example, monitoring fan speed or failure allows software to migrate a computation from a node before it overheats and fails.

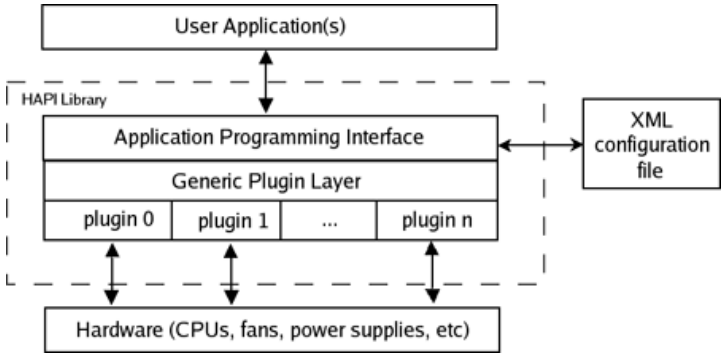


Figure 5 HAPI Fault Tolerance Toolkit

As Figure 5 suggests, the HAPI API supports monitor "plug-ins" for a variety of system-specific sensors, providing standard interfaces for generic access via an XML configuration file. On Linux systems, these typically include temperature readings from the motherboard and CPUs, power supply voltages, disk error rates, and fan speeds. Concretely, HAPI currently supports SMART (Self-Monitoring, Analysis and Reporting Technology) [36] for disks, ACPI [37] for power monitoring and IPMI [38].

Using HAPI, we have monitored the behavior of several Linux clusters when executing differing workloads. These measurements showed the thermal stress imposed by floating point intensive workloads and machine room airflow. Because thermal stress is strongly correlated with long-term hardware failure, understanding these differential stresses can guide batch scheduling and checkpoint/restart frequencies. In our experiments, HAPI also identified improperly configured nodes, where fans were incorrectly installed, suggesting that HAPI can be effective during system acceptance testing. We hope to build on the HAPI toolkit and create an intelligent, adaptive failure monitoring and prediction system that can anticipate impending node failures and shape scheduling and recovery strategies.

3.4. Resilient Workflow Execution

As we discussed earlier, complex workflows executing on large, distributed systems are susceptible to failures at multiple levels (hardware, operating system, middleware, network and application). Hence, deterministic completion guarantees for these workflows are problematic. Instead, successful workflow execution should be viewed as probabilistic, at best.

Given this scenario, we have developed a fault tolerant web service system for resilient workflow execution in the context of the LEAD and VGrADS projects. Our approach relies on a combination of (a) simple restart – application workflow step is restarted on the next best available resource, (b) over-provisioning – multiple copies of

the workflow step are executed and (c) migration – the workflow step is moved to different resources.

The fault tolerance and recovery system (FTR) web service is illustrated in Figure 6, integrated with the LEAD [8] software stack. The service resides between the workflow execution engine (BPEL [18]) and the application service responsible for executing individual workflow steps on remote resources. Hence, the FTR service can control reliable application execution by creating/invoking multiple application services (over-provisioning) or creating/invoking alternate application services (simple restart and migration).

The FTR service selects from among over-provisioning, migration or restart based on an assessment of expected performance and reliability metrics of available resources. The expected application performance is calculated using (a) application performance models on underlying Grid resources, (b) estimates of the communication time needed to transfer data between Grid resources and (c) estimates of queue wait times for accessing large, shared Grid resources.

Our implementation obtains communication and wait time estimates from the Network Weather Service (NWS) [39] and BQP [40] monitoring systems. Because many workflows (e.g., those in LEAD) are deadline driven, our implementation considers the application deadline in the decision process. In this implementation, the user specifies a desired success probability, which is then convolved with performance and reliability estimates to choose an appropriate fault-tolerance mechanism and resource(s) that will satisfy the user-defined success probability and application deadline.

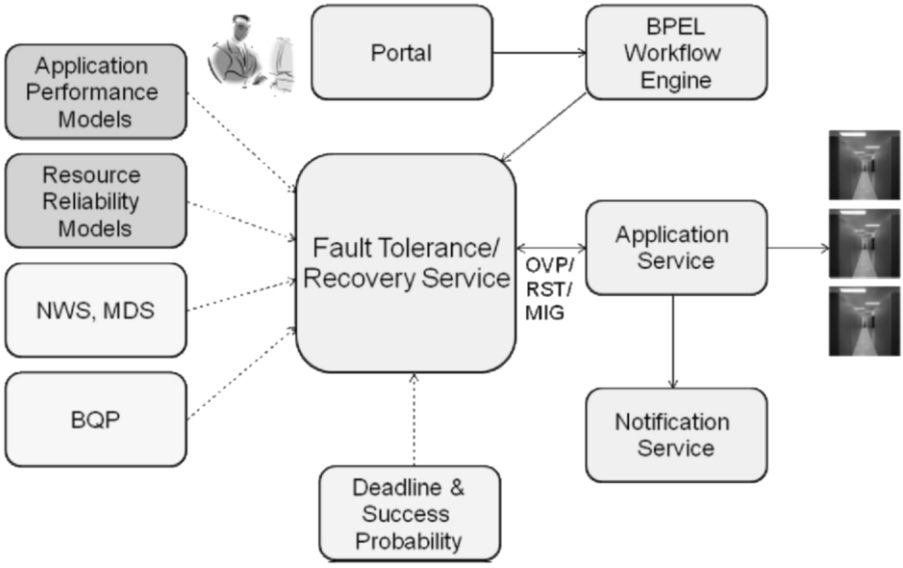


Figure 6 LEAD/VGrADS Fault Tolerance Infrastructure

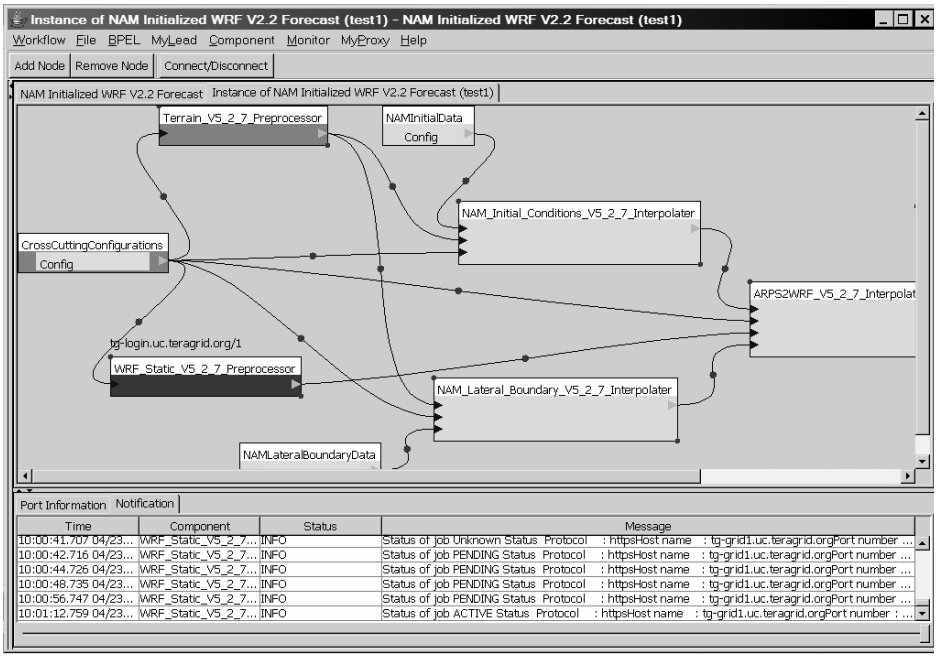


Figure 7 A Resilient LEAD Workflow Execution

Figure 7 illustrates a resilient LEAD workflow execution being monitored via the XBaya workflow composer [41]. A light colored box indicates that a particular workflow step is executing; a dark color indicates that the step has failed. In this example, the "WRF_Static" step has failed, and the FTR service has restarted it on a different resource that satisfies the user success probability and deadline constraints.

4. Complex Workflows and Ensembles

In addition to LEAD's adaptive, mesoscale weather forecasting environment, RENCI collaborates with researchers in biomedicine and disaster response to create and deploy portals and workflows for biomedical discovery and disaster response. Each domain poses a distinct set of application support and integration challenges.

Unlike the physical sciences and engineering, which first embraced workflows for collaborative analysis of experimental data sets from large instruments (e.g., the Grid Physics Network (GriPhyN) [4] or the Large Hadron Collider) and coupling expensive, distributed experimental facilities (e.g., the Network for Earthquake Engineering Simulation (NEES) [42]), the biological and biomedical sciences analyze diverse data sets (often small in size but rich in features) using a large number of software tools. The community is more distributed and has less experience with Grid and web services.

Conversely, disaster response modeling necessarily involves real-time constraints, with potential loss of life and property damage for erroneous predictions. Several programs are run as Monte Carlo ensembles to predict likely outcomes. Although these workflows couple fewer tools (ocean, storm surge, atmosphere and hydrology models) and execute in well-defined ensembles, the data are in real-time streams and the outputs must be relayed to non-technical decision makers.

The use patterns, failure modes and time constraints for the biomedical and disaster response ensembles are strikingly different. Our goal is to generalize insights from these and other workflows to shape development of performance and reliability tools as well as Grid abstractions.

4.1. RENCI TeraGrid BioPortal

The rise of quantitative biology – the application of quantitative methods to analyze observational data and test hypotheses – and the explosive growth of biomedical research are due in large part to increased involvement of multidisciplinary teams, creation of large-scale computational models, mining of distributed data archives and application of high-throughput instrumentation to capture large volumes of biological and biomedical data. The BioPortal [43] is one of a suite of "science gateways" in a nascent cyber-infrastructure [44] supported by the NSF TeraGrid [45] and was motivated by discussions with two distinct groups, (a) molecular biology and (b) genetics and genotype-phenotype analysis.

Drawing on lessons from earlier science gateway projects, the BioPortal was designed for *ease of use*, *scalability*, *extensibility* and *sharing* and builds on many existing tools, including the Open Grid Computing Environment (OGCE) [46], the PISE application description infrastructure [47], the Globus toolkit [48], common

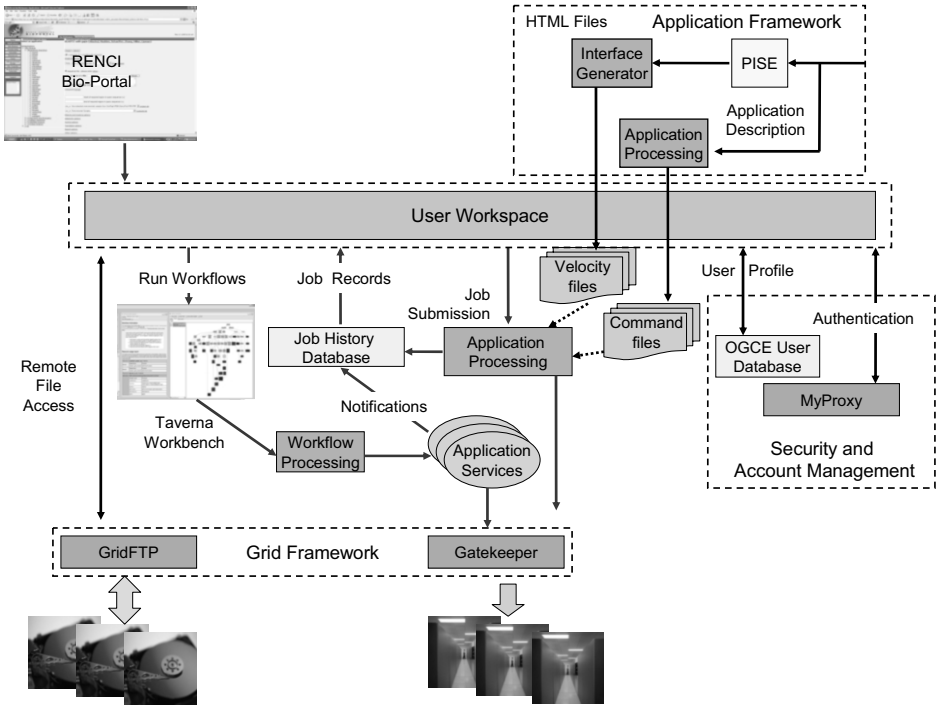


Figure 8 BioPortal Services and Infrastructure

bioinformatics software and databases, and workflow extensions from Taverna [49]. Together, these capabilities allow the BioPortal to

- provide standard access to commonly used bioinformatics data and software,
- use distributed Grid and web services cyber-infrastructure,
- quickly integrate new applications, and
- exploit a scalable, open source software stack.

Figure 8 shows the five primary components of the BioPortal: the user workspace, Taverna workflow specification system, application framework, Grid framework, and security and account management. As the name suggests, the user workspace is the interaction point for user actions with Grid and web services. The Taverna extension allows one to develop and encapsulate workflows as services. In turn, the application framework allows one to describe the interface and processing logic needed to interpret the user's inputs. Finally, the Grid, security and account management components coordinate authentication, job submission and file transfer.

Within the BioPortal, bioinformatics tools and services are accessible at two levels. First, via the PISE framework [47], one can write an XML file describing an application's interfaces and the logic needed to process the inputs. Within the BioPortal, the PISE forms are transformed into a Velocity workflow template which is then used to execute the application. This encapsulation mechanism supports over 140 standard bioinformatics tools and roughly 300 GB of community biological data (predominately genetic sequences).

Although one can use the PISE-encapsulated tools directly, biomedical analysis most often involves a series of processing steps, where the outputs from one tool are inputs to another. Taverna [49] allows users to specify these steps as a workflow, which can then be scheduled and executed atop Grid resources. Figure 9 shows one example of such a workflow, with over 500 steps for each of the thousands of input sequences.

Existing Grid technologies manage jobs and files provided by BioPortal services. Credentials are stored centrally, which OGCE uses for short-lived Globus certificates via MyProxy [50]. With the certificate, each user can access BioPortal services with a single sign-on. In turn, the Globus gatekeeper manages remote job submission, execution and monitoring, and the BioPortal framework manages any data movement required for the inputs and retrieves the outputs.

In addition to supporting RENCI research collaborations, the BioPortal serves as an NSF TeraGrid Science Gateway [45]. The latter provides access to high-end computing systems at TeraGrid resource providers. This required enhancements for improved data staging, community account management and stronger audit tracking for accounting and resource allocation.

Our experiences with the BioPortal exposed many of the software challenges we outlined in §2, namely the importance of reliability for effective use. Because many of the Grid and web services and the biological databases used by BioPortal workflows are distributed worldwide, the probability of successfully executing a workflow declines dramatically as the number of workflow steps rises and the geographic dispersion of step execution increases.

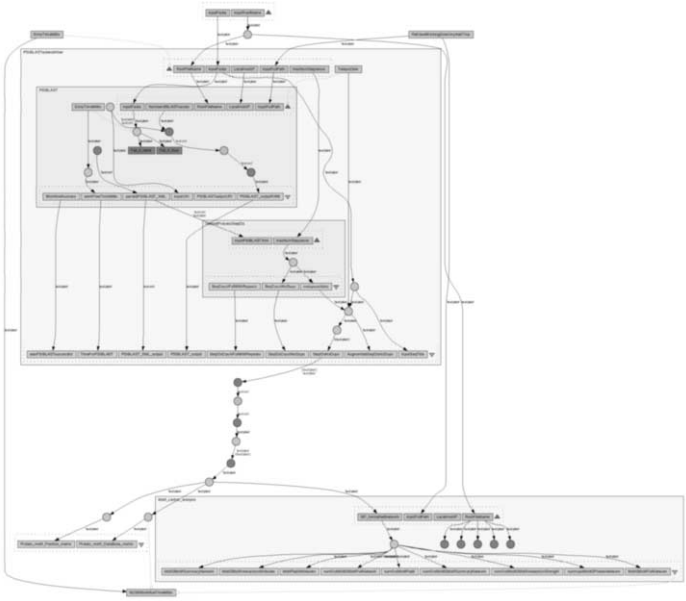


Figure 9 BioPortal TeraGrid Science Gateway

Because some common biological and biomedical workflows involve hundreds of successive steps, fault tolerance mechanisms are not optional. Taverna includes rudimentary retry mechanisms, but these are inadequate for large workflows. One also needs service migration to select new execution sites, over-provisioning to minimize service completion times and monitoring infrastructure to coordinate resilience mechanisms. In addition, data migration for central processing can also reduce failure frequencies. These lessons have influenced our VGrADs research and resilience infrastructure.

4.2. HydroMET Disaster Response

Severe weather is a serious concern worldwide due to its economic and human impacts. In the United States, North Carolina is particularly vulnerable to flooding caused by weather events, from storm surges in coastal regions to rain-triggered inland inundation and flash flooding in steeper terrain. Building on our LEAD project and local expertise, RENCI is collaborating with state and local government to use computational modeling for improved planning and response.

Figure 10 shows a simplified view of the data sources and the computational models used in forecasting coastal flooding. The top row of boxes represents weather data and models. The middle row addresses "over land" flooding sources, and the bottom row captures the storm surge model and the fusion of data and model outputs.

Each of the computational models shown in Figure 10 is a substantial parallel program. An ensemble of runs is needed for each model to address our incomplete

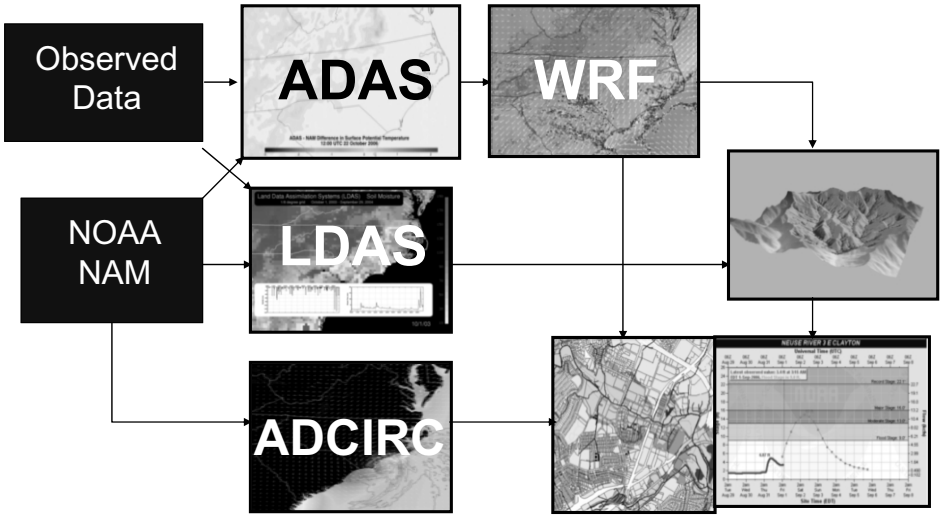


Figure 10 North Carolina HydroMET Workflow

knowledge of the input conditions and other uncertainties. As with LEAD, this complex workflow must be run reliably and under tight deadlines. The HydroMET workflow also shares the property that it must react dynamically to changes in observed data and to produce more detailed predictions in critical areas.

4.3. VGrADS: Virtual Grid Abstractions

The complexity, unreliability and low-level specifications in today's Grids limit their utility and applicability. The Virtual Grid Application Development Software (VGrADS) project attacks a fundamental part of this problem - how to more effectively program these highly complex and dynamic systems. As the project name suggests, VGrADS is based on the concept of grid virtualization, which hides the complexity of Grid resource specification, discovery, scheduling and fault tolerance via simple abstractions.

A virtual Grid Definition Language (vgDL) and a complementary virtual Grid Execution System (vgES) allow users to specify qualitative resource needs and realize those specifications via standard interfaces. The vgDL includes associators that specify the relationships among resources and operators that specify their network connectivity. Hence, rather than requesting a high-performance cluster to execute a WRF weather computation, in vgDL one might request

```
Weather = WRFBag = TightBagOf<CNode>[64:128]; CNode = {memory>=2GB}
```

where TightBagOf<CNode>[64:128] indicates a request for a tightly connected, high bandwidth (TightBagOf) group of processors (nodes) and that any resource with

between 64 and 128 processors is acceptable. In turn, each of the nodes must have at least 2 GB of memory. Notice that this specification does not name a particular resource, nor even require that the set of processors be on the same physical resource. Moreover, the performance targets are qualitative (high connectivity) rather than quantitative (e.g., an Infiniband interconnect).

Given this specification, the vgES is free to provide any resource or resource set that satisfies this specification or even synthesize an appropriate virtual resource from a set of physical resources. To enable applications to guide detailed choices amongst resources to optimize performance, a "ranking function" is available for system use.

The abstraction also enables qualitative reliability specifications, such as [51]

```
RWeather = WRFBag = HighReliabilityBag<NodeSet>[1:1];
NodeSet = TightBagOf<CNode>[64:128]; CNode = {memory>=2GB}
```

where,HighReliabilityBag requires the virtual cluster to have high reliability, without specifying an implementation mechanism. The execution system is then free to either select highly reliable hardware or provide the illusion of reliable hardware by retry, migration, over-provisioning or other mechanisms, as described in §3.4.

With simple recovery specifications and mechanisms, VGrADS, vgDL and vgES enable complex fault recovery and fault avoidance. Combined with the sophisticated scheduling, simplified Grid environments are accessible to a broad range of developers and users.

5. Summary and Future Directions

Scientific computing is moving rapidly from a world of "reliable, secure parallel systems" to a world of distributed software, virtual organizations and high-performance, though unreliable parallel systems with few guarantees of availability and quality of service. For large, complex workflows, high performance and reliability are critical features that must be built into every portion of the application, application framework, libraries and services and operating systems, and also into the hardware. New approaches to performance optimization and software resilience and effective software development and abstraction are critical to the political and technical success of Grids for scientific discovery. Without tools and support at many levels, creating and executing complex, multidisciplinary workflows will be exceptionally difficult if not impossible.

Not only must these tools support parallelism at unprecedented scale and at many levels, libraries, runtime environments and operating systems must be re-architected to provide the required stability. RENCI is pursuing a holistic approach that combines Grid and system fault tolerance and performance optimization research with ongoing application validation via multidisciplinary collaborations.

Acknowledgments

The authors would like to express their thanks to the entire RENCI team for software development, images and insights. Special thanks are due to Kevin Gamiel and Ken Gallupi.

Additional thanks are due to our LEAD and VGrADS collaborators. Support for this work has been partially funded by NSF Linked Environments for Atmospheric Discovery (LEAD) 2003-01685-1, NSF Virtual Grid Application Development Software (VGrADS) R38719-73900004, NSF CyberEval-Collaborative Research: The NSF Cyberinfrastructure Evaluation Center SCI-0510267, DOE National Computational Infrastructure for Lattice Gauge Theory DE-FC02-06ER41445, by DOE PERI, the Performance Engineering Research Center (PERC-3) DE-FC02-06ER25764 and by NIH planning grant RR020751-03.

References

[1] M. Eastwood and K. Cayton, "Quad-Core Processors Bring Higher Performance and Lower Cost to Mainstream Computing," *Intel White Paper*, April 2007

[2] E. Korpela, D Werthimer, D. Anderson, J. Cobb and M. Lebofsky, "SETI@home-Massively Distributed Computing for SETI," *IEEE Multimedia* , Vol. 3, No. 1 pp. 78-93, January 1996

[3] M. Shirts and V. Pande, "Screen Savers of the World Unite!," *Science*, Vol. 290 no. 5498, 8 December 2000

[4] Y. Zhao, M. Wilde, I. Foster, J. Voeckler, J. Dobson, E. Glibert, T. Jordan and E. Quigg, "Virtual Data Grid Middleware Services for Data-Intensive Science," *Concurrency and Computation: Practice and Experience*, Vol 18(6), May 2006

[5] J. Michalakes, J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang, "The Weather Research and Forecast Model: Software Architecture and Performance," *Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing in Meteorology*, October 2004

[6] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "Grid Services for Distributed System Integration," *Computer*, 35(6), 2002

[7] Renaissance Computing Institute, *www.renci.org*, 2007

[8] K. K. Droegemeier, D. Gannon, D. Reed, B. Plale, J. Alameda, T. Baltzer, K. Brewster, R. Clark, B. Domenico, S. Graves, E. Joseph, D. Murray, R. Ramachandran, M. Ramamurthy, L. Ramakrishnan, J. A. Rushing, D. Weber, R. Wilhelmson, A. Wilson, M. Xue and S. Yalda, "Service-Oriented Environments for Dynamically Interacting with Mesoscale Weather," *IEEE Computational Science and Engineering,* Vol. 7, No. 6, pp. 12-29, December 2005

[9] "Leadership-Class System Acquisition - Creating a Petascale Computing Environment for Science and Engineering (program solicitation)," *http://www.nsf.gov/pubs/2006/nsf06573/nsf06573.html*, 2006

[10] J. Dongarra *et al*, "Top500 Supercomputer Sites," *www.top500.org*, 2007

[11] C. W. McCurdy, R. Stevens, H. Simon, W. Kramer, D. Bailey, W. Johnston, C. Catlett, R. Lusk, T. Morgan, J. Meza, M. Banda, J. Leighton and J. Hules, "Creating Science-Driven Computer Architecture: A New Path to Scientific Leadership," *Lawrence Berkeley National Laboratory, LBNL/PUB-5483*, October 2002

[12] D. A. Reed, C-D. Lu and C. L. Mendes, "Reliability Challenges in Large Systems," *Future Generation Computer Systems*, Vol. 22, No. 3, pp. 293-302, February 2006

[13] E. Pinheiro, W. Weber and L.A. Barroso, "Failure Trends in a Large Disk Population," *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST07),* February 2007

[14] C-D. Lu and D. A. Reed, "Assessing Fault Sensitivity in MPI Applications," *Proceedings of SC2004,* Pittsburgh, November 2004

[15] O. Khalili, J. He, C. Olschanowsky, A. Snavely, and H. Casanova, "Measuring the Performance and Reliability of Production Computational Grids," *7th IEEE/ACM International Conference on Grid Computing*, 2006

[16] G. E. Fagg, E. Gaberiel, G, Bosilca, T. Spring, Z. Angskun, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *Future Generation Computer Systems*, Vol 15(5-6), pp. 757-768, 1999

[17] E. N. Elnozahy, L. Alvisi, Y. M. Wang and D. B. Johnson, "A Survey of Rollback-Recovery Protocols in Message Passing Systems," *ACM Computing Surveys*, Vol. 34 issue 3, September 2002

[18] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F.Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic and S. Weerawarana, "Business Process Execution Language for Web Services," *http://www.ibm.com/developerworks/webservices/library/ws-bpel/*, May 2003

[19] M. Feller, I. Foster and S. Martin, "GT4 GRAM: A Functionality and Performance Study," *TeraGrid 2007 Conference*, 2007

[29] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman. "Grid Information Services for Distributed Resource Sharing," *Proceedings of the Tenth International Symposium on High-Performance Distributed Computing*, 2001

[21] A. Bayucan, R. Henderson, C. Lesiak, B. Mann, T. Proett and D. Tweten, "Portable Batch System: External Reference Specification," *MRJ Technology Solutions*, November 1999

[22] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems," *4th Workshop on Job Scheduling Strategies for Parallel Processing*, Springer-Verlag, 1998

[23] I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, "A Security Architecture for Computational Grids," *Proceedings of the 5th ACM Conference on Computer and Communications Security Conference*, 1998

[24] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu and I. Foster, "The Globus Striped GridFTP Framework and Server," *Proceedings of Super Computing 2005 (SC05)*, November 2005

[25] B. Plale, D. Gannon, J. Brotzge, K. Droegemeier, J. Kurose, D. Mclaughlin, R. Wilhelmson, S. Graves, M. Ramamurhty, R. D. Clark, S. Yalda, D.A. Reed, E. Joseph and V. Chandraeskar, "CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting," *IEEE Computer*, Vol. 39, No. 11, pp. 56-64, November 2006

[26] Y. Kee, D. Logothetis, R. Huang, H. Casanova and A. Chien, "Efficient Resource Description and High Quality Selection for Virtual Grids," *Proceedings of the 5th IEEE Symposium on Cluster Computing and the Grid (CCGrid'05)*, 2005

[27] President's IT Advisory Committee (PITAC), *Computational Science: Ensuring America's Competitiveness*, June 2005

[28] C. L. Mendes and D. A. Reed, "Monitoring Large Systems via Statistical Sampling," *The International Journal of High Performance Computing Applications*, Vol.18, No.2, Summer 2004, pp.267-277

[29] T. Gamblin, R. Fowler and D. A. Reed, "Scalable Methods for Monitoring and Detecting Behavioral Classes in Scientific Codes," *in preparation*, 2007

[30] F. Petrini, D. Kerbyson and S. Pakin, "The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q," *Proceedings of SC03*, Phoenix Arizona, November 2003

[31] R. A. Luettich, J. J. Westerink and N. W. Scheffner. "ADCIRC: An Advanced Three-Dimensional Circulation Model of Shelves, Coasts and Estuaries, Report 1: Theory and Methodology of ADCIRC-2DDI and ADCIRC-3DL," *Dredging Research Program Technical Report DRP-92-6, U.S. Army Engineer Research and Development Center*, Vicksburg, MS, 1992

[32] R. A. Luettich and J. J. Westerink, "Theory Report: Formulation and Numerical Implementation of the 2D/3D ADCIRC Finite Element Model Version 43.XX," *http://www.marine.unc/C_CATS/adcirc/adcirc.htm*, March 2003

[33] J. Mellor-Crummey, R. Fowler, G.l Marin, and N. Tallent, "HPCView: A Tool for Top-down Analysis of Node Performance," *The Journal of Supercomputing*, 23:81-104, 2002

[34] N. Froyd, J. Mellor-Crummey and R. Fowler, "Efficient Call-Stack Profiling of Unmodified, Optimized Code," *Proceedings of the International Conference on Supercomputing (ICS2005)*, pages 81-90, Cambridge, MA, June 2005

[35] J. Dongarra, K. London, S. Moore, P. Mucci, D. Terpstra, H. You and M. Zhou, "Experiences and Lessons Learned with a Portable Interface to Hardware Performance Counters," *PADTAD Workshop, IPDPS*, 2003

[36] International Committee on Information Technology Standards, "397-2005(1532d) AT Attachment with Packet Interface - 7"

[37] "Advanced Configuration and Power Interface, *www.acpi.info/*, October 2006

[38] "IPMI - Intelligent Platform Management Interface Specification, Second Generation V2.0," *Intel Hewlett-Packard NEC Dell*, February 2004

[39] R. Wolski, N. Pjesivac-Grbovic, K. London and J. Dongarra. "Extending the MPI Specification for Process Fault Tolerance on High Performance Computing Systems," *International Supercomputing Conference (ICS)* Heidelberg Germany, June 2004

[40] J. Brevik, D. Nurmi and R. Wolski, "Predicting Bounds on Queuing Delay for Batch-scheduled Parallel Machines," *Proceedings of the Eleventh ACM SIGPLAN Symposium on Principle and Practice of Parallel Programming*, 2006

[41] "XBaya: A Graphical Workflow Composer for Web Services," *Indiana University, http://www.extreme.indiana.edu/xgws/xbaya/*, 2007

[42] L. Pearlman, C. Kesselman, S. Gullapalli, B. F. Spencer, J. Futrelle, K. Ricker, I. Foster, P. Hubbard, and C. Severance, "Distributed Hybrid Earthquake Engineering Experiments: Experiences with a Ground-Shaking Grid Application," *Proceedings of the 13th IEEE international Symposium on High Performance Distributed Computing,* pp. 14-23, June 2004

[43] L. Ramakrishnan, M. S.C Reed, J. L. Tilson and D. A. Reed, "Grid Portals for Bioinformatics," *Second International Workshop on Grid Computing Environments (GCE),* Workshop at SC|06, November 2006, Tampa, Florida

[44] Atkins *et al*, "Revolutionizing Science and Engineering Through Cyberinfrastructure," *Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*, January 2003

[45] C. Catlett, "The Philosophy of TeraGrid: Building an Open, Extensible, Distributed Terascale Facility," *Second IEEE/ACM International Symposium on Cluster Computing and the Grid* (*CGRID2002)*, p. 5, 2002

[46] J. Alameda, M. Christie, G. Fox, J. Futrelle, D. Gannon, M. Hategan, G. Kandaswamy, G. V. Laszewski, M. A. Nacar, M. Pierce, E. Roberts, C. Severance, and M. Thomas, "The Open Grid Computing Environments Collaboration: Portlets and Services for Science Gateways," *Concurrency and Computation : Practice & Experience*, Vol 19, 6, pp 921-942, April 2007

[47] C. Letondal, "A Web Interface Generator for Molecular Biology Programs in Unix," *Bioinformatics*, 17(1), pp 73-82, 2001

[48] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, pp 2-13, 2006

[49] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, M. R. Pocock, A. Wipat and P. Li, "Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows," *Bioinformatics*, 20, pp. 3045-3054

[50] J. Novotny, S. Tuecke and V. Welch, "An Online Credential Repository for the Grid: MyProxy," *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10),* August 2001

[51] L. Ramakrishnan and D. A. Reed, "Performability Modeling for Scheduling and Fault Tolerance Strategies for Grid Workflows," *submitted to SC07*, November 2007

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Job Scheduling on the Grid: Towards SLA-Based Scheduling

Rizos SAKELLARIOU [a,1] and Viktor YARMOLENKO [a]
[a] *School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, United Kingdom*

Abstract. This paper argues for the need to provide more flexibility in the level of service offered by Grid-enable high-performance, parallel, supercomputing resources. It is envisaged that such need could be satisfied by making separate Service Level Agreements (SLAs) between the resource owner and the user who wants to submit and run a job on these resources. A number of issues related to the materialization of this vision are highlighted in the paper.

Keywords. Grid Computing, Job Scheduling, Service Level Agreement

1. Introduction

Traditionally, the scheduling mechanisms used for the enactment of jobs on parallel supercomputer resources have been queue-based. Such mechanisms are essentially offering only one level of service, which can be summarized simply as 'run the job when it gets to the head of the queue' (even though, in some cases, some jobs that are behind in the priority queue may be assigned to unutilized processors, to ensure that there are no idle processors, using a technique known as backfilling [1,2]).

The emergence of Grid Computing [3] has created new opportunities to support compute and/or data intensive scientific applications, which, among other, may have large computational resource requirements. However, at the same time, Grid computing is built upon the notion of *virtual organizations* [4]. This is a group of individuals and/or institutions who collaborate towards the solution of a particular problem through a set of resource sharing rules. The key implication of this *coordinated resource sharing* [4] is that the collaboration of the members of a Virtual Organization may span across different administrative domains. In the context of scheduling independent jobs of a large, parallelizable application that makes use of the Grid, such a collaboration, across multiple administrative domains, would also require some coordination across the different schedulers of the individual resources employed. Lack of coordination may annul all the benefits from parallelism that one might expect when independent jobs are running, in parallel, onto different resources.

To illustrate this problem, consider an application that can be represented by a Directed Acyclic Graph (DAG). Different nodes of the graph represent individual jobs

[1]Corresponding Author: Rizos Sakellariou, School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, United Kingdom; E-mail: rizos.sakellariou@manchester.ac.uk.

with computational requirements, whereas edges represent communication of data. In the context of Grid computing, DAGs can be used to represent many applications that belong to the important family of applications collectively known as *scientific workflows* [5,6,7,8,9,10,11]. In order to exploit the task parallelism available in the DAG (by executing independent nodes in parallel), let us assume that two different nodes are assigned to different resources, each resource belonging to a different administrative domain and using its own scheduler with its own queueing system. Since each job will start execution when it reaches the head of the local queue (of the corresponding resource), it is possible (because of different waiting times) that one job may finish execution before the other job has even started. From a global point of view, this is equivalent to executing the two jobs sequentially. In other words, there are no performance benefits due to the exploitation of parallelism because of the different behaviour of the queue-based schedulers of the local resources, which have acted without coordination and contrary to user expectations.

Advance Reservation of resources has been suggested as a means to guarantee that tasks will run onto a resource when the user expects them to run [12,13]. Essentially, advance reservation specifies a precise time that jobs may start running. This allows the user to request resources from systems with different schedulers for a specific *time interval* (e.g., start time, finish time), thereby obtaining a sufficient number of resources for the time s(he) may need. Advance reservation has already received significant attention and has been considered an important requirement for future Grid resource management systems [14]. There has been already significant progress on supporting it by several projects and schedulers, such as the Load Sharing Facility platform (LSF) [15], Maui [16], COSY [17], and EASY [18,19].

However, advance reservation is again an extreme level of service since it specifies a precise time when jobs can be made to run. This may cause several problems to the resource owner, and there is some scepticism in the community, especially with respect to the degree to which advance reservations contribute to improving the overall performance of a scheduler [20]. For example, when an advance reservation is made, the scheduler must place jobs around this fixed job. Typically, this is done using backfilling [2], which increases utilisation by searching the work queues for small jobs to plug the gaps. In practice, this rarely works perfectly, and so the scheduler must either leave the reserved processing elements empty for a time, or suspend or checkpoint active jobs near to the time of the reservation. These processes are not instantaneous; e.g., checkpointing a 64 processor Unified Weather Model job on an O3800 takes about 12 minutes, despite a small total memory footprint of 3Gb; checkpointing 256 processor jobs can exceed one hour. Suspension is faster, but can adversely affect the performance of the incoming job, due to the cost of swapping out memory used by the suspended job when it is required by the incoming job. Either way, there are gaps in the schedule, i.e., CPU time which is not processing users' work. As utilisation often represents income for the service owner, there is a tendency to offset the cost of the unused time by charging for advance reservation jobs at a considerably higher tariff. While it is possible to set tariffs high enough to compensate, this brute-force solution is inefficient in terms of resources, and undesirable for both users, who pay higher prices, and for resource owners, who must charge uncompetitive prices and watch utilization fall.

In recent years, there has been work aiming to explore the space between the two aforementioned extreme levels of service, namely, 'run this job whenever it gets to the head of the queue', or 'run this job at this precise time'. The main idea is to provide differ-

ent levels of service by forging agreements between the different parties (user, resource owner, etc). Such agreements are agreed on the basis of different constraints expressed by the user and/or the resource owner and essentially specify a desired (and agreed) level of service. The use of *Service Level Agreements* (SLAs) gives rise to a fundamentally new approach for job scheduling on the Grid.

This paper provides an overview of the issues surrounding the design of a fundamentally new infrastructure for job scheduling on the Grid, which is based on the notion of Service Level Agreements (SLAs), based on work carried out as part of a recently completed project [21]. In the most general form, such SLAs are negotiated between a client (user, superscheduler, or resource broker) and a provider (the owner of a resource with its own scheduler), may contain information such as acceptable job start and end times, and may be re-negotiated during runtime.

Some background and an overview of relevant work on SLAs as well as the ways that they can be used on the Grid is given in Section 2. Section 3 describes the key features of the envisaged architecture for job scheduling using SLAs on the Grid. Section 4 describes the key issues and challenges that need to be addressed for the materialization of such an architecture. Sections 5 and 6 elaborate on approaches to address some of these issues, namely the description of the terms in an SLA as well as the design of heuristics for SLA scheduling, building on earlier work presented in [22,23,24]. Finally, Section 7 concludes the paper.

2. Background and Related Work

An SLA can be described as a legally binding contract between the parties involved. The agreement relates to a transaction for the provision of a service; as a result, the parties of an SLA can be distinguished between providers and consumers of a service. The terms of the SLA describe the expected level of service within which the service will be provided; these have been agreed between service providers and service consumers.

It has been mentioned that forms of SLAs were in operation since the 1960s, "when they were used as a method for buying minutes of computer machine time" [25]. In more recent years, SLAs became more widespread as a means to make agreements when outsourcing IT functions [26], or when providing network services [27,28,29]. Most of these agreements were paper-based and were drawn after some form of negotiation between appropriate persons.

The shifting emphasis of the Grid towards a service-oriented paradigm — as well as trends in application service delivery to move away from tightly coupled systems towards structures of loosely coupled, dynamically bound systems [30] — led to the adoption of Service Level Agreements as a standard concept by which work on the Grid can be allocated to resources and enable coordinated resource management. In the context of Grid and Web services, the current understanding of the community is that such an SLA is essentially an electronic contract, which is expected to be negotiated fully automatically (i.e., without any human intervention) by different processes and, as such, much be machine readable and understandable.

As a result, there has been a significant amount of research, in recent years, on various topics related to SLAs. Issues related to their overall incorporation into grid architectures have been discussed in [31,32,33,34,35]. Issues related to the specification

of the SLAs have been considered in [36,37,38,39]. Issues specifically related to (and motivated from) the usage of SLAs for resource management on the Grid have been considered in [34,40,41,42,43,44,45]. Of particular importance has been the work on the negotiation of SLAs. Since the early influential work on the Service Negotiation and Acquisition Protocol (SNAP) [41], further work has argued for the need to take into account the principles of contract law when negotiating SLAs in order to form legally binding agreements [46], even considering how the consequences from the use of SLAs may need to be taken into account by guidelines on electronic contracts [47]. Other work has examined issues related to trust and security [48,49], or targeted more business oriented case studies [50]. Later work has also drawn upon relevant research carried out particularly in the context of agents [51,52], where several techniques have been used to model negotiation, ranging from heuristics to game theoretic and argumentation-based approaches [53]. Finally, a significant area of research relates to the economic aspects associated with the usage of SLAs for service provision (e.g., charges for successful service provision, penalties for failure, etc.); relevant work has been presented in [54,55, 56,57,58,59].

Work within the Grid Resource Allocation Agreement Protocol (GRAAP) Working Group of the Open Grid Forum [60] (and earlier within its predecessor, the Global Grid Forum) has led to the development of *WS-Agreement (WS-A)* [61], a specification for a simple generic language and protocol to establish agreements between two parties. Each of the two parties can be either an initiator of or a responder to the agreement. The agreement structure is composed of several distinct parts, namely Name, Context and Terms of Agreement. The latter is also divided in service description terms and guarantee terms. Service descriptions terms mainly describe the functionality to be delivered under the agreement. The guarantee terms define the assurance on service quality for each item mentioned in the service description terms section of the WS-A. In the specific context of job submission, which is the focus of this paper, such assurances may be defined as a parameter (constant) or bounds (min/max) on the availability of part or the whole of the resource. In WS-A, such assurances are referred to as service level objectives (SLOs); in a domain specific to computation services provision, they are usually expressed as values (e.g., SLO: CPUcount = 8). Each SLO may refer to one or more business values, called a business value list (BVL). This list expresses different value aspects of a specific SLO. The other two types of guarantee terms are Qualifying Conditions and Importance, which have a similar function to SLO and BVL, respectively.

3. An Architecture for Job Scheduling on the Grid Using Service Level Agreements

The key vision of this paper is that jobs, submitted for execution to high-performance computing resources, are associated with an SLA. This SLA is negotiated between a client (e.g., a user or a resource broker) and a provider (the owner of a resource with its own local scheduler) and contains information about the level of service agreed between the two parties, such as acceptable job start and end times.

An overall architectural view, materializing this vision, is presented in Figure 1. There are three key 'players' underpinning this materialization. *Users* negotiate and agree an SLA with a resource broker (or superscheduler, or coordinator). *Brokers* nego-

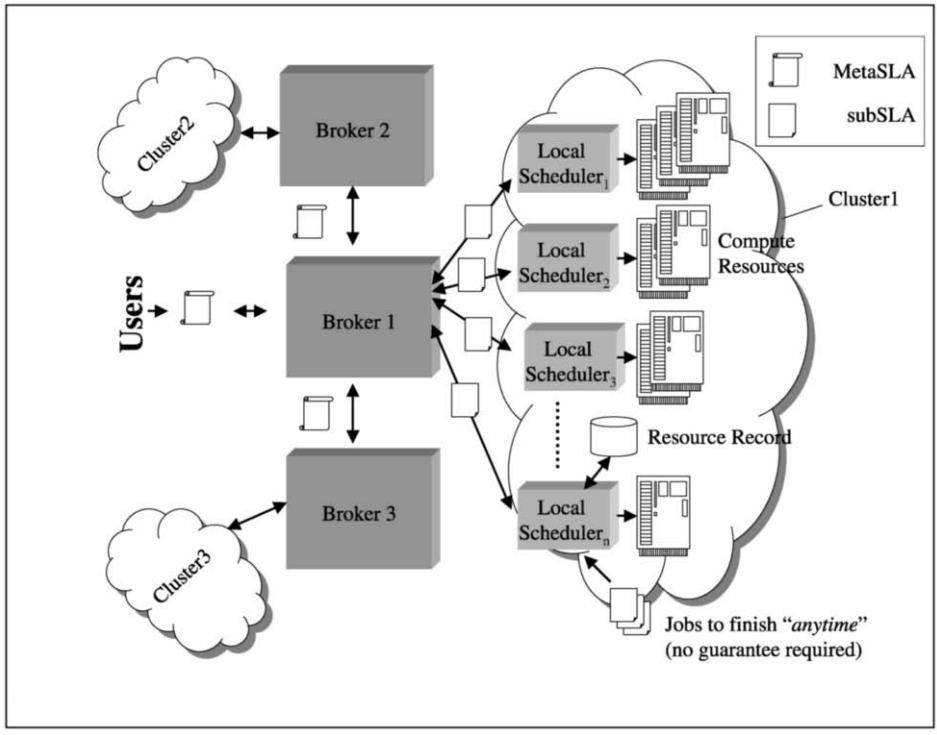


Figure 1. An overview of an architecture for SLA-based job scheduling.

tiate and agree an SLA with users; these SLAs may be mapped to one or more SLAs, which are negotiated and agreed with local resources and their schedulers. Finally, *local schedulers* need to schedule the work that is associated with an SLA which they agreed to (the constraints associated with such an SLA, agreed by a resource, may be stored locally in the resource, in some kind of a resource record). It is also noted that a single SLA agreed between a user and a broker may 'translate' to multiple SLAs between the broker and different local resources to serve the user's request (for example, this could be the case when the SLA between a user and a broker refers to a workflow application with several tasks that are executed on different resources. In such case, the user may want to set constraints for the workflow as a whole and the broker may have to translate it to specific SLAs for individual tasks, following an approach similar to the one described in [62]); to indicate the possible differences between these two types of SLA, the terms *meta-SLA* and *sub-SLA* are used. Furthermore, as indicated in the figure, this SLA-based view for job submission, may still allow the submission of jobs that are not not associated with an SLA; however, no guarantees about their completion time would be offered in this case.

It should be stressed also that the primary objective of the architectural view presented in Figure 1 is to illustrate the fundamentally different interactions that may arise as a result of the use of SLAs to make an agreement on the level of service expected when jobs submitted. It is beyond the scope of this paper to argue for or against a particular type of architecture. For example, one may assume that a broker is associated

with a single cluster (as in the figure), within the same administrative domain, or with many, across different administrative domains. Similarly, the broker may be more tightly connected to the local schedulers (in which case, the broker may exercise more control on the local schedulers and no SLA between the broker and the local schedulers may be necessary), or no SLAs may be agreed between different brokers. However, besides the pros and cons of different architectural choices, there are several common challenges that need to be addressed; this is the main focus in the remainder of this paper.

4. Issues and Challenges with SLA-Based Job Scheduling

This section groups some of the key challenges that need to be addressed in order to materialize an SLA-based vision for job submission and scheduling. These challenges could be summarized as follows:

SLA vocabulary: The vision of SLA based scheduling assumes that the SLAs themselves are machine readable and understandable. This implies that any agreements, between the parties concerned, for a particular level of service need to be expressed in a commonly understood (and legally binding) language. There has been early work on generic languages for SLA description [38,39], but none related to the particular problem of the requirements associated with job submission and execution (on possibly high-performance computing resources). Some of the issues that may arise in such cases, and some of the trade-offs in the description of the vocabulary are discussed in the next section.

Negotiation: It is envisaged that SLAs may be negotiated between machines and users or only between machines. In this negotiation some commonly agreed protocol needs to be followed. This protocol needs to take into account both the nature of the distributed systems and networks which are used for the negotiation (for example, what if an offer from one party is not received by the other party due to a network failure), and should abide by appropriate legal requirements (for example, should the receipt of every proposal for an agreement be acknowledged or not?). In addition, during negotiation, machines should be able to reason about whether an offer is acceptable and possibly they should be able to make counter-offers. As already mentioned in Section 2 there has been a significant amount of work on these topics. However, the relevant challenges on negotiation are more of a generic nature rather than specific to the problem of using SLAs for job scheduling.

Scheduling: Given that, currently, scheduling of jobs on high-performance compute resources is mostly based on priority queues (with the possible addition of backfilling techniques [1,2]), the use of SLAs would require the development of a new set of algorithms for efficient scheduling, which would be based on satisfying the terms agreed in the SLA. The existence of efficient scheduling algorithms would be of paramount importance to estimate capacity and reason on the possible acceptance (by a local resource) of a new request to make an SLA. Some approaches related to the development of such algorithms are discussed further later on.

Constitutional Aspects: In the context of any SLA based provision, sooner or later, the need for dispute resolution may arise. In addition, users may also be interested in the reliability of specific brokers; for example, how likely (or unlikely) is that a broker will honour an SLA (even if breaking the SLA would require the broker to pay a penalty). This issue of modelling reputation may also be related to the approaches followed for pricing and/or penalties when agreeing SLAs (for example: is there a flat charge for the usage of resources? do the fees vary depending on particular circumstances?). Such issues, might become more important as the envisaged usage of SLA-based job scheduling grows. However, since relevant work is in its infancy, simple assumptions, without neglecting the challenges that might arise later in this respect, could be made for a start.

A more detailed overview of the issues and trade-offs involved with particular choices with respect to the first and the third challenge above is given in the following two sections.

5. Issues on the Expressiveness of SLAs

Experience from trying to incorporate SLAs in distributed grid computing has indicated that this task is not as easy as defining a dictionary of terms, which are then placed in some kind of an XML derived structure. For example, work within the Open Grid Forum (OGF) to define an SLA specification, has lasted for years and there have been several debates about what an SLA should contain and/or how it should look like. For instance, one of the difficulties that had to be addressed related to how issues related to the SLA document description would be decoupled from issues related to the negotiation protocols.

It must be noted here, that an efficient SLA related environment, with guarantees for a certain level of fault tolerance, may often include renegotiation [27,31,41] of the whole or part of an SLA. In the context of SLA-based job scheduling, simulation studies [44] have indicated that, at high resource load, a significant proportion of the agreed SLAs may have to be renegotiated in order to avoid failure of an SLA. Such renegotiation may have an overhead and may require the user's participation at one or more stages between the time when the initial agreement was made and when the work, specified in the SLA, is completed. It would clearly undermine the overall vision of Grid, as an environment based on the service-oriented paradigm and ultimately driven by principles similar to those of autonomic systems [63,64], to require a possibly repeated user involvement every time renegotiation is in order. One would expect some abilities of self-management in this respect.

The trade-off (and the dilemma), here, is whether a much simpler SLA description (which, however, may cause more often the need for renegotiation) is more (or less) desirable than a more complex SLA description, which, however, enables the system to deal more effectively (and possibly transparently from the user) with the need for renegotiation. In other words, simplicity in the SLA description itself at the expense of renegotiation overhead, or more efficient scheduling of the SLAs [44,24] at the expense of more complex SLA descriptions? The idea first presented in [22,44] attempts to answer this question by making SLAs (infinitely) more *expressive*. Such increased expressiveness in SLAs is achieved by describing SLA terms as arbitrary *analytical functions*, as opposed to the traditional approach of specifying SLAs [61] that uses constant values or ranges

for SLA terms. The authors in [22,44] have argued that the encapsulation of SLA terms in analytical functions can potentially reduce the need for renegotiation of a service and provide the extra flexibility needed for a wider spectrum of resource management decisions. The motivation, an example and some issues related to this approach to make an SLA more expressive will be described next.

In most cases of SLA-based resource management applications [33,65], the set of guarantee terms is rigidly defined. Within such an SLA, it is known in advance, for example, what the exact financial gain will be when an SLA is audited. There could be reasons to believe that this arrangement is not flexible and carries too little information for the provider bound by this SLA to perform its tasks successfully without the need for renegotiation. The expressive capacity of an agreement can be improved by adding more terms, but there is a practical limit on the SLA size, whilst most of the terms could easily be described analytically. The latter description can provide an infinitely large set of term configurations in SLA. Analytical functions can describe complex relationships between SLA terms that could potentially be used by autonomic applications in providing qualitatively new levels of service, described in an SLA with a higher degree of expressiveness. Such an SLA cover an infinitely large number of (agreed and acceptable) outcomes that belong to the continuum defined by the system of analytical functions.

To illustrate all this, we use an example from [23]; this relates to the reservation of resources for a computational job where using a different number of parallel processors to run the job would still result in a successful SLA from the user point of view. Naturally, the running time of such a (parallel) job would vary, depending on the parallelism assigned to it by the scheduler (that is, the number of processors assigned). The use of analytical functions means that the scheduler can now schedule and repeatedly reschedule this job in order to meet the objectives of its associated SLA without the need for renegotiation. Thus, the SLA terms relating to the size of the job (with respect to execution time needed and how this may relate to CPUs used) may look like:

- Maximum number of CPU Nodes that can be used, N_{CPU}^{max}
- CPU Nodes reserved, $N_{CPU} = \{2, 3, 4, ..., N_{CPU}^{max}\}$
- Reserved time for job execution, $t_D = \frac{t_D^1}{N_{CPU}}$

where t_D^1 is the projected time for the job to complete if it runs on a single CPU Node; and N_{CPU}^{max} is limited by the capacity of the resource. It can be seen that the reserved time for job execution is described as a function of the number of CPU nodes reserved. If a function was not used, then the SLA would have to include a single value for each of N_{CPU} and t_D; any attempt to deviate from these values would trigger the renegotiation of the SLA.

Of course, the relation between t_D and N_{CPU} is, in reality, more complex than what is described in the function used above. However, the point to make is that, even in this simple case, where SLA terms describe a limited number of configurations (N_{CPU}^{max} to be precise), the number of possible renegotiations needed would be prohibitively high, had all this been described in the traditional way as just a list of terms. Simulation shows that, even in such a simple case, the negotiation overhead per job decreases dramatically when the description in the SLA makes use of functions to link terms. It also suggests that by describing more SLA terms as functions, it is possible to achieve maximum utilisation with minimum negotiation. Moreover, novel scheduling algorithms may now be

deployed, something that was not possible before, precisely because of the renegotiation cost. In [23], more examples of using an SLA with variable network bandwidth or resource load are presented; including some consideration of pricing terms.

The main argument that can be made against this approach (of describing SLAs in a more expressive manner) is the perceived complexity of the agreement. Indeed, for the SLA terms that are defined by very complex functions that in turn depend on other SLA terms, the agreement itself becomes a system of equations that may suffer from discontinuities, may have holes in solutions or may have no solution at all. Such representations of SLA terms could potentially put a very heavy strain on clients and providers, who would have to verify the SLA, check for contradictions, run away arguments and other unpleasantries. However, many of the problems would be reasonably easy to detect already and the complexity of the SLA could grow gradually with the readiness of environment, without any changes to SLA itself. Furthermore, a conservative approach could also be applied when negotiating SLAs. As in real life, if one party cannot understand all the terms in a proposed agreement, there is no obligation to agree.

Another important point made in [23], which might be overlooked easily, relates to the so-called set of *universal* terms. These are terms whose value is not known at the time an SLA is formed; they can be combined with analytical functions to make the SLAs more expressive. The most obvious argument in favour of the universal terms relates to the ability to enable agreements on terms (to form an SLA), the status of which may not be known at the time of making the agreement; yet, keeping the SLA in a concise form. To give an example, the actual amount of network traffic that a job will incur may not be known at the time an SLA is formed. Still, the existence of a universal term for network traffic could be used, for instance, in the earlier example as part of the analytical function describing the reserved time for job execution.

The use of universal terms makes it possible to create more speculative agreements, which encompass more possible outcome scenarios and uncertainties in a single concise SLA document, reducing further the need to renegotiate the agreement in the case of a failure (that has happened or is anticipated). To give an example, one such *universal* term, wall-clock time, was used in [24] to agree on the price for the service. The users were prepared to pay a higher price if the job was executed sooner rather than later. Hence, the price term, agreed as a function of the wall clock time (and other terms), enabled the SLA to cover an infinite number of possible outcomes with respect to the time constraints. This, in turn, enabled the scheduler to make more intelligent decisions in maximising profit for its owner, since the total revenue depending on different schedules could be calculated and taken into account. Such analysis would not be practical with traditional SLAs (that is, SLAs that do not make use of functions as described here to allow for more expressiveness), because the scheduler would have to renegotiate the pricing with hundreds of users.

6. A Review of Issues on Scheduling SLAs

Despite the growing interest in Service Level Agreements, there has been surprisingly little research published on the topic of using the information contained in the (agreed) SLAs for planning and scheduling purposes (with the objective of satisfying the requirements of the SLAs successfully). In the context of SLA-based job scheduling, the very

idea of using SLAs is to alter the approach used to perform job scheduling. This means that local schedulers of high-performance computing resources must now take into account not only functional properties of the submitted job (such as parallelism and execution time) but also non-functional requirements expressed as terms and constraints in SLA. Even though the existences of various constraints that need to be satisfied may point to a constraint satisfaction problem, standard methods based on various forms of (exhaustive) searching would not be practical in a grid environment where a lage number of SLAs may have to be negotiated and scheduled quickly. Simple scheduling heuristics could provide efficient scheduling solutions with negligible time overheads for a large set of SLAs; a number of different scheduling heuristics was investigated in [24].

The principle behind these heuristics is based on how SLAs would be prioritised. Scheduling, then, falls to a simple routine of picking jobs (in order of priority) from the prioritised list and fitting them onto the resource, without rescheduling previous jobs already allocated – a single iteration packing process.

The priority value, H, is computed using a function, whose generic form is as follows:

$$H = \min(h_1 + w \cdot h_2) \tag{1}$$

$$H = \max(h_1 + w \cdot h_2) \tag{2}$$

where h_1,h_2 is one of:

- Earliest job start time, T_S
- Latest job finish time, T_F
- Reserved time for job execution, t_D
- Number of CPU Nodes required, N_{CPU}
- Job size, measured in CPU-hours, $A = N_{CPU} \times t_D$
- Job laxity, defined as $t_L = T_F - (T_S + t_D)$

and w is a weighting coefficient that can be both positive and negative. Obviously, for each heuristic, h_1 and h_2 are different, so the total number of heuristics that can be created combining all the terms above with each other is 15. By sweeping across values of w in equations 1 and 2 the best effort configuration can be found for each heuristic.

To test these SLA-aware scheduling heuristics the following scenario and SLA template were used. In a simple model, which consists of a *Client*, a *Provider* and an *Agreement* (SLA) between the two, the requests from clients (bound by an SLA) were scheduled by the local resource scheduler; the latter used different heuristics for the local scheduling of these requests. The SLA in this case consisted of the following five terms: (i) Earliest job start time, T_S; (ii) Latest job finish time, T_F; (iii) Reserved time for job execution, t_D; (iv) Number of CPU Nodes required, N_{CPU}; and (v) Final price agreed, V_{TOT}. It was assumed that two different pricing policies were used to calculate the income for the Provider:

- *flat rate*: The charge for each SLA is the same, regardless of duration of the job, number of CPUs used, etc.
- *pay as you go*: The charge is proportional to the actual usage of the resource (that is, the product of the time and the number of CPUs used).

heuristic	SLA%	CPU%	w
$\min(T_F + wN_{CPU})$	97.0%	84.8%	0.48
$\min(T_F + wA)$	96.8%	82.8%	0.06
$\min(T_S + wA)$	96.1%	83.0%	0.28
$\min(T_F + wt_D)$	95.4%	88.0%	-0.025
$\min(T_S + wt_D)$	95.4%	87.0%	10.0
$\min(T_F + wt_L)$	95.3%	86.3%	0.015
$\min(T_S + wt_L)$	95.3%	86.0%	1.15
$\min(T_F + wT_S)$	95.3%	85.5%	-0.05
$\min(T_S + wN_{CPU})$	92.7%	77.5%	4.5

Table 1. Evaluation of the performance of different SLA scheduling heuristics using a flat rate pricing policy.

heuristic	SLA%	CPU%	w
$\min(T_F + wt_D)$	92.0%	94.0%	-6.63
$\min(T_S + wt_L)$	92.2%	93.9%	0.4
$\min(T_F + wT_S)$	91.9%	93.9%	2.0
$\min(T_F + wt_L)$	92.1%	93.7%	-0.7
$\min(T_S + wt_D)$	91.7%	93.7%	3.2
$\min(T_S + wN_{CPU})$	85.0%	93.3%	-0.63
$\min(T_F + wA)$	89.7%	93.2%	-0.3
$\min(T_S + wA)$	89.0%	92.8%	0.0
$\min(T_F + wN_{CPU})$	90.5%	89.3%	-3.0

Table 2. Evaluation of the performance of different SLA scheduling heuristics using a flat rate pricing policy.

Nine of the fifteen heuristics that could be created following the approach described above, were evaluated. The choice of these nine heuristics was based on experience from results in [24]. Thus, the nine heuristics refer to all combinations that include at least one of the terms T_F and T_S. For the evaluation, we used a simple (synthetic) SLA workload model whereby, for each SLA workload set generated, a scheduling solution existed that resulted in a 100% resource utilisation for the SLAs in the set. For each of the two pricing policies, 100 SLA workload sets were generated for a single experiment. The results then were averaged over 100 experiments. On average, each SLA workload consisted of around 380 SLAs. The latter were supposed to be scheduled on a homogeneous parallel resource of 48 CPU nodes, which was available for 400 hours, creating thus a reference frame of 48×400 in size. Each of the different SLA workloads was scheduled on the resource of the same size using all nine different heuristics.

The averaged results are shown in Table 1 (for the 'flat rate' pricing policy) and in Table 2 (for the 'pay as you go' pricing policy). The results show, for each heuristic considered (first column): the average percentage of successfully scheduled SLAs out of all the SLAs that could be scheduled in principle (SLA satisfiability – column 2); the average percentage of hours the CPU was busy out of the total number of hours available (CPU utilization — column 3); and the value of w that was chosen to maximize income for each pricing policy (column 4). In each table, the results are sorted starting from the heuristic that provides on average the highest income (clearly, in the case of the 'flat rate'

pricing policy the income is proportional to SLA satisfiability, while in the case of the 'pay as you go' pricing policy the income is proportional to CPU utilization).

Our experiments revealed several interesting points:

- Depending on the pricing policy used it was possibly to achieve either high SLA satisfiability (over 95% in the case of the 'flat rate' pricing policy) or high CPU utilization (over 89% in the case of the 'pay as you go' pricing policy). It is noted that these are average values; in individual cases (especially on a small-scale reference frame), these percentages could be even higher. These are very encouraging results considering the SLA scheduling problem, especially if we take into account that scheduling was performed in a fraction of a second.
- The performance of the heuristics appears to be related to the pricing policy used. Thus, heuristics that achieve the highest percentage of SLA satisfiability using the 'flat rate' pricing policy do not seem to achieve the highest percentage of CPU utilization using the 'pay as you go' pricing policy and *vice versa*. This observation is consistent with the results in [24] where it was found that the choice of the pricing policy can change dramatically the effect of different heuristics. On average, the heuristic based on T_F and T_D appears to perform best in both cases.
- Optimizing for CPU utilization (as in Table 2) appears to provide more robust results in the sense that the level of SLA satisfiability is not affected as much as CPU utilization is affected when the objective is to optimize for SLA satisfiability (as in Table 1).
- When the job laxity reaches $\approx 0.5 \cdot 10^2$ of t_D the time constraints of the job appear to no longer affect the performance of any of the considered scheduling heuristics.
- On average, when the ratio of the average value of N_{CPU} to the size of the resource was, roughly, 1:8 most heuristics seemed to result in the highest utilisation.
- The performance of some heuristics was very sensitive to the workload. For example heuristics based on N_{CPU} and A change in a very nontrivial way with the change of the distribution of CPU requests.

It must be said here that this was a limited study. Although it indicated that heuristics could be used to provide good solutions to the scheduling problems, additional studies, with varying workloads are highly desirable.

7. Conclusion

This paper argued for the need of a fundamental approach for job scheduling on (parallel) high-performance computing resources, based on service level agreements. This approach appears to become more pertinent with developments in the context of Grids, service-oriented architectures and autonomic computing. The paper highlighted a number of issues that need to be addressed in order to materialise such a novel approach. Some work aiming to address some of these aspects, in particular related to the description of the SLA terms and their use in scheduling, was briefly presented. The authors' view is that SLA based approaches for resource provision are highly promising; however, there is a need for further advances and research in the challenges indicated before such approaches become common-place.

Acknowledgements

The research presented in this paper has been funded by EPSRC (Grant Reference: GR/S67654/01); their support is gratefully acknowledged. Support from the CoreGRID Network of Excellence is also acknowledged.

References

[1] Dror G. Feitelson and Ahuva M. Weil. Utilization and predictability in scheduling the IBM SP2 with backfilling. In *Proceedings of the 12th International Parallel Processing Symposium*, pages 542-546, April 1998.

[2] David A. Lifka. The ANL/IBM SP Scheduling System. *Proceedings of the IPPS'95 Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, volume 949, pp. 295-303, 1995.

[3] Ian Foster and Carl Kesselman (eds). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.

[4] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3), pp. 200-222, 2001.

[5] The Workshop on Workflows in Support of Large-Scale Science (WORKS06), in conjunction with HPDC2006, Paris, 2006. http://www.isi.edu/works06

[6] The 2nd Workshop on Workflows in Support of Large-Scale Science (WORKS07), in conjunction with HPDC2007, Monterey Bay, California, 2007. http://www.isi.edu/works07

[7] Jim Blythe, Sonal Jain, Ewa Deelman, Yolanda Gil, Karan Vahi, Anirban Mandal, and Ken Kennedy. Task scheduling strategies for workflow-based applications in grids. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, pp. 759-767, 2005.

[8] Anirban Mandal, Ken Kennedy, Charles Koelbel, Gabriel Marin, John Mellor-Crummey, Bo Liu, and Lennart Johnsson. Scheduling Strategies for Mapping Application Workflows onto the Grid. In *IEEE International Symposium on High Performance Distributed Computing (HPDC 2005)*, 2005.

[9] Arun Ramakrishnan, Gurmeet Singh, Henan Zhao, Ewa Deelman, Rizos Sakellariou, Karan Vahi, Kent Blackburn, David Meyers, and Michael Samidi. Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, 2007, pp. 401-409.

[10] Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Schields. *Workflows for e-Science. Scientific Workflows for Grids*. Springer, 2007.

[11] Marek Wieczorek, Radu Prodan, and Thomas Fahringer. Scheduling of Scientific Workflows in the ASKALON Grid Environment. In *SIGMOD Record*, volume 34(3), September 2005.

[12] Jon MacLaren. Advance Reservations: State of the Art. In Global Grid Forum 9 (GGF9), Scheduling and Resource Management Workshop, Chicago, USA, October 2003.

[13] Warren Smith, Ian Foster, and Valerie Taylor. Scheduling with Advanced Reservations. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, pages 127-132, May 2000.

[14] Uwe Schwiegelshohn, Philipp Wieder, and Ramin Yahyapour. Resource Management for Future Generation Grids. In *Future Generation Grids*, Vladimir Getov, Domenico Laforenza, Alexander Reinefeld (Eds.), Springer, CoreGrid Series, 2005.

[15] Load Sharing Facility platform. *http://www.platform.com/products/LSF/.*

[16] David B. Jackson, Quinn Snell, and Mark J. Clement. Core Algorithms of the Maui Scheduler. In *Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Lecture Notes in Computer Science, volume 2221, pp. 87-102, 2001.

[17] Junwei Cao and Falk Zimmermann. Queue Scheduling and Advance Reservation with COSY. In *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS'04)*, Santa Fe, USA, April 2004.

[18] David A. Lifka, Mark W. Henderson, and Karen Rayl. Users Guide to the Argonne SP Scheduling System. Technique Report ANL/MCS-TM-201, Argonne National Laboratory, May 1995.

[19] Joseph Skovira, Waiman Chan, Honbo Zhou, and David A. Lifka. The EASY – LoadLeveler API Project. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, volume 1162, pp. 41-47, 1996.

[20] Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Parallel Job Scheduling — A Status Report. In *Revised Selected Papers from the 10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2004)*, Lecture Notes in Computer Science, volume 3277, pp. 1-16, 2004.

[21] Service Level Agreement Based Scheduling Heuristics. EPSRC funded project, GR/S67654/01.

[22] Rizos Sakellariou and Viktor Yarmolenko. On the Flexibility of WS-Agreement for Job Submission. In *Proceedings of the 3rd International Workshop on Middleware for Grid Computing (MGC'05)*, Grenoble, France, November 28 - December 02, 2005, ACM International Conference Proceedings Series, vol. 117, 2005.

[23] Viktor Yarmolenko and Rizos Sakellariou. Towards Increased Expressiveness in Service Level Agreements. *Concurrency and Computation: Practice and Experience*, 19(14), 25 September 2007, pp. 1975-1990.

[24] Viktor Yarmolenko and Rizos Sakellariou. An Evaluation of Heuristics for SLA Based Parallel Job Scheduling. In *Proceedings of the 3rd High Performance Grid Computing Workshop (in conjunction with IPDPS 2006)*, Rhodes, Greece, April 2006, IEEE Computer Society Press.

[25] Jill Dixon and Melany Blackwell. Service Level Agreements – A framework for assuring and improving the quality of support services to faculties. In *Proceedings of the 2002 Annual International Conference of the Higher Education Research and Development Society of Australasia (HERDSA)*, 2002.

[26] Matthew K. O. Lee. IT Outsourcing contracts: Practical Issues for Management. *Industrial Management and Data Systems*, 1996.

[27] M. D'Arienzo, A. Pescapè, S. P. Romano, and G. Ventre. The service level agreement manager: control and management of phone channel bandwidth over Premium IP networks. In *Proceedings of the 15th International Conference on Computer Communication (ICCC02)*, pages 421–432, Washington DC, USA, 2002. International Council for Computer Communication.

[28] Dinesh Verma. *Supporting Service Level Agreements on IP Networks*, Macmillan Technical Publishing, 1999.

[29] Dimitrios Kagklis, Nicolas Liampotis, and Christos Tsakiris. Architecture for the Creation of Service Level Agreements and Activation of IP Added Value Services. In *Proceedings of the 7th International Conference on Telecommunications (ConTEL2003)*, pp. 689-692, 2003.

[30] A. Keller, G. Kar, H. Ludwig, A. Dan, and J. L. Hellerstein. Managing dynamic services: a contract based approach to a conceptual architecture. *Network Operations and Management Symposium (NOMS2002)*, pp. 513-528, 2002.

[31] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. *Agreement-based Grid Service Management (OGSI-Agreement)*. Global Grid Forum, GRAAP-WG Author Contribution, 12 Jun, 2003.

[32] Asit Dan, Catalin Dumitresku, and Matei Ripeanu. Connecting Client Objectives with Resource Capabilities: An Essential Component for Grid Service Management Infrastructures. In *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC04)*, 2004.

[33] D.G.A. Mobach, B.J. Overeinder, and F.M.T. Brazier. A resource negotiation infrastructure for self-managing applications. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing (ICAC 2005)*, Seatle, WA, 2005.

[34] Vijay K. Naik, Swaminathan Sivasubramanian, and Sriram Krishnan. Adaptive Resource Sharing in a Web Services Environment. In *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, Lecture Notes in Computer Science, vol. 3231, pp. 311-330, 2004.

[35] James Padgett, Mohammed Haji, and Karim Djemame. SLA Management in a Service Oriented Architecture. In *Proceedings of the International Conference on Computational Science and its Applications (ICCSA2005)*, Lecture Notes in Computer Science, volume 3483, pp. 1282-1291, 2005.

[36] Alexander Keller and Heiko Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11(1), March 2003, pp. 57-81.

[37] Heiko Ludwig, Alexander Keller, Asit Dan, Richard King, and Richard Franck. A Service Level Agreement Language for Dynamic Electronic Services. *Electronic Commerce Research*, 3(1-2), January 2003, pp. 43-59.

[38] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P. King, and Richard Franck. *Web Service Level Agreement (WSLA) language specification*. IBM Corporation, 2003.

[39] James Skene, D. Davide Lamanna, and Wolfgang Emmerich. Precise Service Level Agreements. *Proceedings of the 26th International Conference on Software Engineering (ICSE2004)*, pp. 179-188.

[40] Lars-Olof Burchard, Matthias Hovestadt, Odej Kao, Axel Keller, and Barry Linnert. The Virtual Resource Manager: An Architecture for SLA-aware Resource Management. In *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid (CCGrid2004)*, pp. 126-133.

[41] Karl Czajkowski, Ian Foster, Carl Kesselman, Volker Sander, and Steven Tuecke. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In *Proceedings of the 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Edinburgh, Scotland, UK, July 24, 2002, Lecture Notes in Computer Science, vol. 2537, pp. 153-183.

[42] Catalin L. Dumitrescu and Ian Foster. GRUBER: A Grid Resource Usage SLA Broker. In *Proceedings of Euro-Par 2005*, Lecture Notes in Computer Science, vol. 3648, pp. 465-474.

[43] Catalin Dumitrescu, Ioan Raicu, Ian Foster. DI-GRUBER: A Distributed Approach to Grid Resource Brokering In *Proceedings of the 2005 ACM/IEEE Supercomputing (SC 2005)*, November 12-18, Seattle, Washington, USA.

[44] Viktor Yarmolenko, Rizos Sakellariou, Djamila Ouelhadj and Jonathan M. Garibaldi. SLA Based Job Scheduling: A Case Study on Policies for Negotiation with Resources. In *Proceedings of the UK e-Science All Hands Meeting 2005 (AHM'2005)*, Nottingham, UK, 19-22 September 2005, CD-ROM Proceedings (Editors: Simon J. Cox, David W. Walker).

[45] Avraham Leff, James T. Rayfield, and Daniel M. Dias. Service-level agreements and commercial grids. *IEEE Internet Computing*, 7(4), July-August 2003, pp. 44-50.

[46] Michael Parkin, Dean Kuo, and John Brooke. A Framework & Negotiation Protocol for Service Contracts. In *Proceedings of the 2006 International Conference on Services Computing (SCC06)*, Chicago, USA, 18-22 September 2006, pp. 253-256.

[47] Michael Parkin, Dean Kuo, John M. Brooke, and Angus MacCulloch. Challenges in EU Grid Contracts. In *Proceedings of eChallenges e-2006 Conference*, Barcelona, Spain, 25-27 October 2006, pp. 67-75.

[48] Sakyibea Darko-Ampem and Maria Katsoufi. Towards A Secure Negotiation Protocol for Virtual Organisations. Master of Science Thesis, Department of Computer and Systems Sciences, Stockholm's University / Royal Institute of Technology, May 2006.

[49] TRUSTCOM EU project.

[50] Peer Hasselmeyer, Henning Mersch, Bastian Koller, H.-N. Quyen, Lutz Schubert, Philipp Wieder. Implementing an SLA Negotiation Framework. *Proceedings of e-Challenges 2007*.

[51] Reid G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, Vol. C-29(12), December 1980, pp. 1104-1113.

[52] Pu Huang and Katia Sycara. A Computational Model For Online Agent Negotiation. In *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS35'02)*, IEEE Computer Society Press, 2002.

[53] N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, M.J. Wooldridge and C. Sierra. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, 10(2), March 2001, pp. 199-215.

[54] Alexander Barmouta and Rajkumar Buyya. GridBank: a Grid Accounting Services Architecture (GASA) for distributed systems sharing and integration. In *Proceedings of the 2003 International Parallel and Distributed Processing Symposium (IPDPS'03)*, 22-26 April 2003, IEEE Computer Society Press.

[55] Steven Newhouse, Jon MacLaren, and Katarzyna Keahey. Trading Grid services within the UK e-science Grid. In *Grid resource management: state of the art and future trends* (Editors: Jarek Nabrzyski, Jennifer M. Schopf, Jan Weglarz), Kluwer Academic Publishers, pp. 479-490, 2004.

[56] D. Grosu and A. Das. Auctioning resources in Grids: model and protocols. *Concurrency and Computation: Practice and Experience*, vol. 18(15), 2006, pp. 1909-1927.

[57] Torsten Eymann, Michael Reinicke, Werner Streitberger, Omer Rana, Liviu Joita, Dirk Neumann, Björn Schnizler, Daniel Veit, Oscar Ardaiz, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, Michele Catalano, Mauro Gallegati, Gianfranco Giulioni, Ruben Carvajal Schiaffino, and Floriano Zini. Catallaxy-based Grid markets. *Multiagent and Grid Systems*, vol. 1(4), 2005, IOS Press, pp. 297-307.

[58] Liviu Joita, Omer F. Rana, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, and Oscar Ardaiz. Application Deployment on Catallactic Grid Middleware. *IEEE Distributed Systems Online*, 7(12):0612–oz001, 2006.

[59] Zhen Liu, Mark S. Squillante, and Joel L. Wolf. On Maximizing Service Level Agreement Profits. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pp. 213-223, 2001.

[60] Open Grid Forum. http://www.gridforum.org

[61] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. *Web Services Agreement Specification (WS-Agreement)*, version 2005/09. Proposed Recommendation to the Global Grid Forum, 20 September 2005.

[62] Henan Zhao and Rizos Sakellariou. Advance Reservation Policies for Workflows. In *Proceedings of the 12th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, June 2006, Saint-Malo, France. Lecture Notes in Computer Science, volume 4376, pages 47-67.

[63] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1), pp. 41-50, January 2003.

[64] R. Sterritt and M. G. Hinchey. Why Computer-Based Systems Should Be Autonomic. In *Proceedings of the 12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2005)*, pp. 406-414, Greenbelt, MD, USA, 3-8 April 2005.

[65] Martyn Fletcher, Jim Austin, and Tom Jackson. Distributed Aero-Engine Condition Monitoring and Diagnosis on the Grid: DAME. In *Proceedings of the 17th International Congress on Condition Monitoring and Diagnostic Engineering Management (COMADEM2004)*, Cambridge, UK, 23-24 Aug, 2004.

Chapter 3

Grid Technology

This page intentionally left blank

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications

Charlie CATLETT[a], William E. ALLCOCK[a], Phil ANDREWS[b], Ruth AYDT[f], Ray BAIR[a], Natasha BALAC[b], Bryan BANISTER[b], Trish BARKER[f], Mark BARTELT[k], Pete BECKMAN[a], Francine BERMAN[b], Gary BERTOLINE[c], Alan BLATECKY[j], Jay BOISSEAU[d], Jim BOTTUM[l], Sharon BRUNETT[k], Julian BUNN[k], Michelle BUTLER[f], David CARVER[d], John COBB[e], Tim COCKERILL[f], Peter F. COUVARES[o], Maytal DAHAN[d], Diana DIEHL[b], Thom DUNNING[f], Ian FOSTER[a], Kelly GAITHER[d], Dennis GANNON[g], Sebastien GOASGUEN[l], Michael GROBE[q], Dave HART[b], Matt HEINZEL[a], Chris HEMPEL[d], Wendy HUNTOON[h], Joseph INSLEY[a], Christopher JORDAN[b], Ivan JUDSON[a], Anke KAMRATH[b], Nicholas KARONIS[m,a], Carl KESSELMAN[n], Patricia KOVATCH[b], Lex LANE[f], Scott LATHROP[a], Michael LEVINE[h], David LIFKA[s], Lee LIMING[a], Miron LIVNY[o], Rich LOFT[i], Doru MARCUSIU[f], Jim MARSTELLER[h], Stuart MARTIN[a], Scott MCCAULAY[g], John MCGEE[j], Laura MCGINNIS[h], Michael MCROBBIE[g], Paul MESSINA[a,k], Reagan MOORE[b], Richard MOORE[b], J.P. NAVARRO[a], Jeff NICHOLS[e], Michael E. PAPKA[a], Rob PENNINGTON[f], Greg PIKE[e], Jim POOL[k], Raghu REDDY[h], Dan REED[t], Tony RIMOVSKY[f], Eric ROBERTS[d], Ralph ROSKIES[h], Sergiu SANIELEVICI[h], J. Ray SCOTT[h], Anurag SHANKAR[g], Mark SHEDDON[b], Mike SHOWERMAN[f], Derek SIMMEL[h], Abe SINGER[b], Dane SKOW[a], Shava SMALLEN[b], Warren SMITH[d], Carol SONG[c], Rick STEVENS[a], Craig STEWART[g], Robert B. STOCK[h], Nathan STONE[h], John TOWNS[f], Tomislav URBAN[d], Mike VILDIBILL[b,r], Edward WALKER[d], Von WELCH[f], Nancy WILKINS-DIEHR[b], Roy WILLIAMS[k], Linda WINKLER[a], Lan ZHAO[c] and Ann ZIMMERMAN[p]

[a] *University of Chicago/Argonne National Laboratory;* [b] *San Diego Supercomputer Center;* [c] *Purdue University;* [d] *Texas Advanced Computing Center;* [e] *Oak Ridge National Laboratory;* [f] *National Center for Supercomputing Applications;* [g] *Indiana University;* [h] *Pittsburgh Supercomputing Center;* [i] *National Center for Atmospheric Research;* [j] *University of North Carolina;* [k] *California Institute of Technology;* [l] *Clemson University;* [m] *Northern Illinois University;* [n] *University of Southern California Information Sciences Institute;* [o] *University of Wisconsin;* [p] *University of Michigan;* [q] *Indiana University – Purdue University Indianapolis;* [r] *Sun Microsystems;* [s] *Cornell University;* [t] *Microsoft*

Abstract. TeraGrid is a national-scale computational science facility supported through a partnership among thirteen institutions, with funding from the US National Science Foundation [1]. Initially created through a Major Research Equipment Facilities Construction (MREFC [2]) award in 2001, the TeraGrid facility began providing production computing, storage, visualization, and data collections services to the national science, engineering, and education community in January 2004. In August 2005 NSF funded a five-year program to operate, enhance, and

expand the capacity and capabilities of the TeraGrid facility to meet the growing needs of the science and engineering community through 2010. This paper describes TeraGrid in terms of the structures, architecture, technologies, and services that are used to provide national-scale, open cyberinfrastructure. The focus of the paper is specifically on the technology approach and use of middleware for the purposes of discussing the impact of such approaches on scientific use of computational infrastructure. While there are many individual science success stories, we do not focus on these in this paper. Similarly, there are many software tools and systems deployed in TeraGrid but our coverage is of the basic system middleware and is not meant to be exhaustive of all technology efforts within TeraGrid. We look in particular at growth and events during 2006 as the user population expanded dramatically and reached an initial "tipping point" with respect to adoption of new "grid" capabilities and usage modalities.

Keywords. Grids, distributed computing, computational science, infrastructure, high-performance computing

Introduction

The TeraGrid[1] facility is an integrated portfolio of more than twenty high-performance computational (HPC) systems, several specialized visualization resources and storage archives, and a dedicated continental-scale interconnection network. *Policy and planning* integration allows the national user community to request access through a single national review process and use the resources of the facility with a single allocation. *Operational and user support* integration enables the user community to interact with many distinct resources and HPC centers through a common service, training, and support organization – masking the complexity of a distributed organization. *Software and services* integration creates a user environment and standard service interfaces that lower barriers to porting applications, enable users to readily exploit the many TeraGrid resources to optimize their workload, and is catalyzing a new generation of scientific discovery through distributed computing modalities.

The TeraGrid mission is to advance science through three integrated initiatives:

- **Deep: Enable Terascale/Petascale Science**: TeraGrid will enable scientists to pursue scientific discovery through an integrated set of Terascale resources and services.
- **Wide: Empower Communities**: TeraGrid will make Terascale resources and services broadly available through partnerships with community-driven service providers.
- **Open: Provide an Extensible Foundation for Cyberinfrastructure**: TeraGrid will provide, and use where provided by others, a set of foundational services and resources to support nation-wide cyberinfrastructure, using open standards, policy, and processes.

The user community that relies on this national facility has dramatically expanded, from under 1,000 users in October 2005 to over 4,000 users at the close of 2006. Nearly 2000 of these are new users with development allocations to explore TeraGrid, port their codes, and incorporate HPC services into their science (§1.1).

[1] The "TeraGrid" project name, chosen in 2001, now more appropriately describes the individual resources, however the aggregate capacity of TeraGrid computing resources was over 700 Teraflops by early 2008 and will exceed one Petaflops by the end of 2008.

TeraGrid resources are also growing exponentially. In early 2006 the largest capability computing resources within TeraGrid were 10–15 Teraflops and ~2,000 processors. By the end of 2007 the largest resource, an NSF-funded system at TACC, will be over 500 Teraflops and 60,000 processor cores, and similar scale systems are planned for 2008, 2009, and 2010 [3]. Storage systems are also growing significantly. Thus, the TeraGrid team is beginning a multi-year challenge to work with the user community to provide training, porting, and optimizing support in order to fully exploit this fundamentally new scale of capability moving into the Petascale regime, while continuing to support a growing user community accessing dozens of resources nation-wide.

The multi-level integration of the TeraGrid facility is also enabling new usage modalities – and corresponding new user communities – that harness HPC, storage, visualization and data resources through advanced software applications and services, often through web portals (§1.2). The introduction of TeraGrid-wide distributed computing building blocks (§2) such as information services, remote job submission, single sign-on, parallel file transfer, and workflow support – in part based on emerging web services technologies – has catalyzed a set of discipline-specific, community-provided "Science Gateways" (§3.2). Gateways interact with TeraGrid resources through these services and related policies to serve communities of 100's to 1000's of scientists and educators.

Enabling Petascale science, supporting the increasing number of new users, and the growth in adoption of new usage modalities and science gateways all require a coordinated approach to building and sustaining a workforce that can fully realize the promise of cyberinfrastructure. Our user support and operations teams leverage the expertise across the TeraGrid partner institutions (§3) and our education, outreach, and training work is focused on a comprehensive set of programs – "HPC University" (§3.3).

The TeraGrid facility and organizational model consists of a set of independent, cooperating resource providers (RPs) working together with a Grid Infrastructure Group (GIG), which facilitates coordination, software and service integration, operations, management, and planning [4]. The GIG is a distributed team with staff located at multiple TeraGrid RP sites as well as other partner institutions. TeraGrid governance borrows from concepts developed in other types of organizations such as open source software projects and standards bodies, which harness the efforts and creativity of many independent participants. Policy and key decisions regarding all aspects of the TeraGrid facility are developed and approved through a forum comprised of representatives from each of the RPs and from the GIG. Results of these decisions are recorded in a persistent document series with a record of consensus among representatives.

In this paper we provide an overview and analysis of TeraGrid in four major sections. First we examine TeraGrid resources and usage as of late 2006, with an analysis of usage modalities and growth. We then turn to three aspects of TeraGrid integration: software and services, user and operational support, and policy and planning.

1. Resources, Usage, and User Community Analysis

TeraGrid resources include computational, storage, and visualization systems as well as specialized data collections and information management capabilities. TeraGrid RP sites are interconnected via a dedicated, optical, wide-area network with individual site connections ranging from 10 Gb/s to 30 Gb/s (Fig. 1). Each computational resource

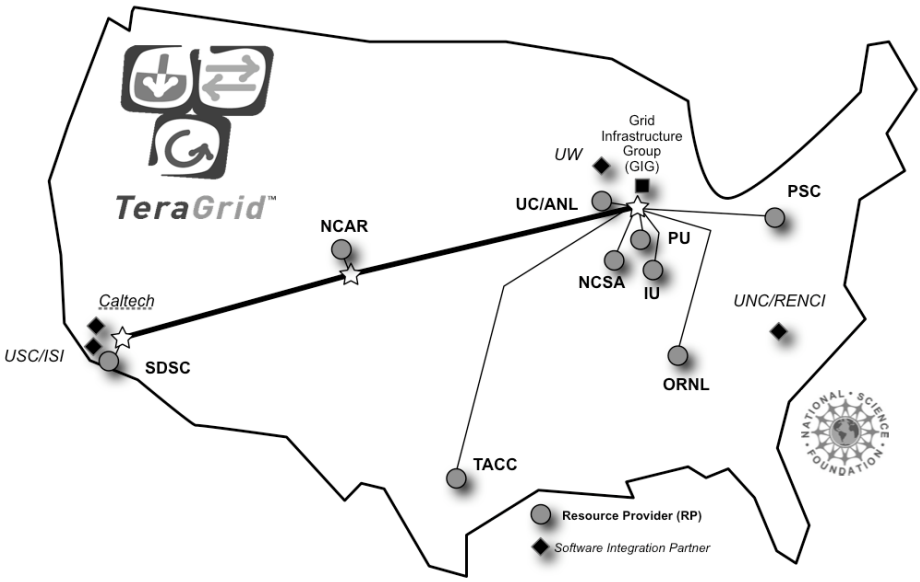


Figure 1. TeraGrid Partner Sites.

Table 1. TeraGrid Resource Growth from 2004 to 2006

Resources	2004	2005	2006
HPC Resources	8	16	23
Storage Resources (supporting allocations)	3 (0)	6 (0)	9 (3)
Data Collections	40	90	101
Science Gateways	0	11	20

provides, in addition to traditional local user environments and services, a set of coordinated TeraGrid software and services that enable TeraGrid-wide capabilities such as single sign-on, common allocations and accounting, or advanced features such as workflow. We begin with an overview of the portfolio of resources, the policies and structures that grant access to users, and an analysis of the user community and their use of the TeraGrid facility. Subsequent sections will detail the software and services, user and community support and engagement, and overall TeraGrid organization.

1.1. Overview of TeraGrid Resources and Usage

Many metrics can be used to examine the adoption of a given set of resources and services and their impact on the science and engineering research and education. In Table 1 we summarize four key growth metrics: resource portfolio, allocation awards granted, resource usage, and the size and nature of the user base. In this section we examine these four metrics.

1.1.1. Resource Portfolio

During 2006 the number of TeraGrid HPC computational resources increased from 16 to 23, expanding aggregate computational capacity by a factor of 2.5, from roughly 120

to nearly 300 Teraflops. TeraGrid's largest computational resources as of early 2007 range from 50–100 Teraflops, with the smallest under 5 Teraflops. The overall computational portfolio includes nearly every type of system architecture, microprocessor, and operating system available for high-performance computing.

TeraGrid HPC systems are owned and operated, in most cases, by the local resource provider institution. Some of the systems were purchased using NSF awards while others were purchased with local institutional or other funding sources. The resources are provided to the national community, through the TeraGrid facility, based on cooperative agreements between the resource providers and NSF. These agreements specify funding levels for resource and user support as well as the percentage of the resource that will be made available through the national allocations process (described below in §1.1.2).

Five TeraGrid RP sites provide storage archives supporting long-term data management. Together these archives currently hold approximately 10 Petabytes of user data, up 50% since 2005.

Users access resources through available wide area Internet access including commercial as well as Internet2 and National Lambda Rail national backbone networks. Interconnection of TeraGrid resources themselves employs a dedicated nationwide optical backbone network with hubs in Los Angeles, Denver, and Chicago. RPs are responsible for maintaining connectivity to other TeraGrid resources to support high-performance data transfer and resource interaction (e.g. workflow), typically through one or more 10 Gb/s connections to a TeraGrid network hub.

A high-performance wide-area parallel filesystem supports tight coupling between TeraGrid resources at three sites using IBM's GPFS-WAN [5] and harnessing TeraGrid's dedicated optical network. Over 500 Terabytes of storage at SDSC can be mounted for remote file I/O from SDSC, ANL, and NCSA. In 2007 several additional RPs will begin offering this service and multiple TeraGrid RP sites are experimenting with alternative wide area distributed filesystems such as Lustre [6]. As of early 2007, filesystems at Indiana University, based on Lustre, can be mounted from TeraGrid systems at NCSA, PSC, and ORNL.

Increasingly, users are also interested in making data collections available in standardized fashion to enable their use in grid applications, workflow, etc. There are over 100 data collections available through TeraGrid, and in 2006 it became clear that a common set of information would be needed in order to ensure that users could readily find and access data collections. We developed a set of minimum requirements for TeraGrid data collections that includes a general description and information about data provenance, access mechanisms, and necessary metadata. Based on these requirements [7] we maintain a data collections directory within the TeraGrid User Portal.

1.1.2. Peer-Reviewed Access to TeraGrid Resources and Services

Users access TeraGrid based on allocation awards made through a national peer-review process. The resource allocation committee (RAC) is a rotating team of several dozen computational scientists, from a variety of disciplines, serving 2- to 3-year terms. RAC members are nominated by TeraGrid resource provider sites, with input from program officers at the National Science Foundation.

Eligibility for TeraGrid use is limited to researchers (post-doctoral included) or educators at U.S. academic or non-profit research institutions. A qualified advisor may

apply for an allocation for his or her class, but high school, undergraduate, and graduate students may not be principal investigators (PIs).

Review criteria are related to the computational requests and appropriate resource use, rather than a review of the underlying scientific theory or research objectives. As proposals are typically associated with a funded research program, the scientific peer review has already taken place and is not repeated by the TeraGrid RAC.

The allocations process has traditionally been focused primarily on computational resources, but was expanded to include major storage requests in 2006. In 2007 the process will be expanded further to facilitate input from the RAC regarding the allocation of dedicated support staff to assist specific projects (see §3.1). This process will involve requests for assistance in units of FTE-months. The RAC will rank these requests and these rankings will be used as input to the user support staffing allocations decisions made by management at TeraGrid RP sites.

Computational Allocation Awards

Computational allocations are measured in Service Units, or "SUs." TeraGrid SUs translate to CPU hours on a given resource based on a "normalized unit" (NU) which is the equivalent of one hour of CPU time on a Cray X-MP. The relative performance rating for a given resource is taken by scaling performance on the HPL [8] benchmark (a standard component of determining the Top 500 rankings [9]). A rough conversion for modern processors is 1 SU = 20 NU.

Proposals are grouped into three categories based on the number of SUs requested. Large requests (currently defined as >500 k SUs) are reviewed semi-annually and medium requests (30–500 k SUs) are reviewed quarterly. Allocations are granted for periods of 12 months, with extensions to 18 months upon request. Small requests (<30k SUs) are reviewed on an ongoing basis by an internal TeraGrid review team using the same qualifying criteria. These requests are generally new projects exploring how TeraGrid resources may help the project's scientific goals, and often involving benchmarking and code porting or for classroom instruction.

In addition to tracking allocations by overall size of award, as shown in Table 2, there are two types of allocations granted for TeraGrid computational resources:

- ***Specific*** allocations are tied to a particular TeraGrid resource, at a particular site.
- ***Roaming*** allocations are usable on any TeraGrid compute resource.

Users may request either of these types of allocations or a combination of the two. Many projects also have multiple *specific* type allocations for different machines. To illustrate the overall mix of *specific* versus *roaming* allocations for large and medium sized awards, we examine the 88 medium allocations in detail in Table 3. All development awards are *roaming* allocations.

Most TeraGrid allocation awards involve a principal investigator (PI) and a small group of collaborators and/or students. However, some awards are used to support larger communities of dozens or even hundreds of users through science gateways (§3.2). We refer to these as *community allocations*.

Storage Resource Allocation Awards

Traditionally, access to storage archives has not been regulated for TeraGrid users – any project has been permitted to store as much data in tape archives as required for their project without special arrangement. Due to sustained, exponential growth in stor-

Table 2. TeraGrid Allocations, Usage, and User Community Growth from 2005 to 2006

Allocations	2005	2006	% Growth
Large proposals awarded (new)	62 (13)	88 (22)	42% (69%)
Medium proposals awarded (new)	70 (50)	160 (92)	129% (84%)
Dev. proposals awarded (new)	123 (115)	229 (209)	86% (82%)
Active TeraGrid PIs	361	1,019	182%
Usage			
NUs Requested	1.3 B	2.96 B	128%
NUs Awarded	844 M	1.92 B	127%
NUs Available (max)	881 M	2.23 B	153%
NUs Delivered	565 M	1.28 B	127%
NUs used by TG Staff	10.4 M	10.1 M	−3%
Jobs run	594,756	1,686,686	184%
Users (Total)			
Users with accounts during 2006	1,712	4,190	145%
Users charging jobs during 2006	876	1,731	98%
Users with accounts on 31-Dec	1,468	3,126	113%
User Institutions (charging jobs)	151	265	75%
US states (charging jobs)	37	47	27%
Users by Allocation Size			
Large Users (# charging jobs)	509 (238)	1,152 (496)	126% (108%)
Medium Users (# charging jobs)	542 (248)	1,087 (423)	101% (71%)
Dev. Users (# charging jobs)	661 (365)	1,948 (783)	195% (115%)

Table 3. Breakdown of allocations for the 88 large (>500 k SU) projects

	No roaming	Roaming
Multiple resources	53	4
Single resources	26	0
No specific allocation	0	5

age for those projects dealing with very large data sets and the associated costs of managing this data, the TeraGrid team evaluated storage costs, trends, and user storage requirements in 2006 [10]. Based on this analysis, the TeraGrid peer-review allocations process now requires proposals from projects that anticipate needs for long-term tape storage above a threshold defined as a function of computational allocation. Requests for tape storage independent of a compute allocation can also be requested. Similarly, projects requiring long-term dedicated space on the TeraGrid wide area parallel file system (GPFS-WAN) obtain this space through the peer-review process.

1.2. Analysis of Technology Usage Patterns of the TeraGrid User Community

During the allocation proposal process each PI designates a discipline area, selecting from a menu of NSF science divisions. Usage by discipline is summarized in Fig. 3. Note that while there are 10 disciplines that account for 94% of TeraGrid usage, there are 20 disciplines that collectively consume the remaining 6%. At the same time these 20 disciplines account for nearly 30% of the overall user population.

As shown in Table 2, overall TeraGrid HPC computational resource usage grew by over a factor of two (127%) during 2006, and the number of jobs executed grew by nearly a factor of 3 (184%). The large increase in jobs can be attributed in part to in-

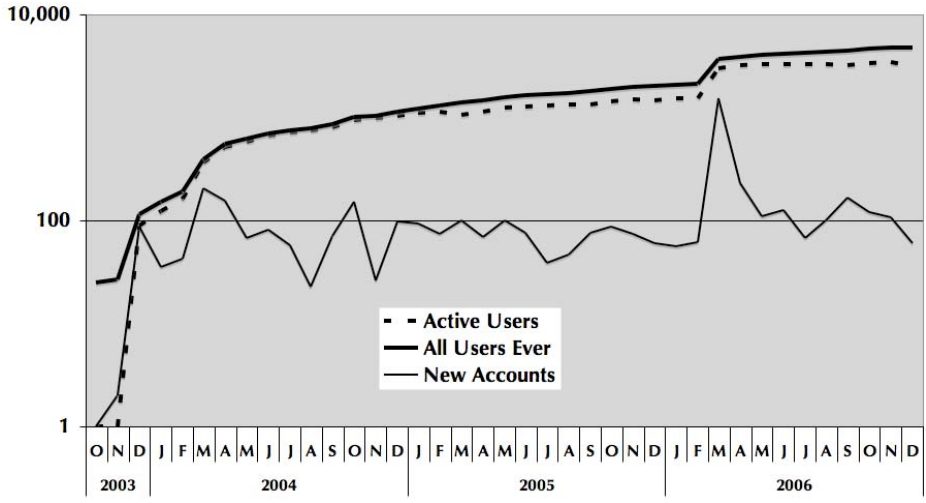


Figure 2. The TeraGrid user population grew by 2800 users during 2006, including a one-time addition of order 1500 users when NCSA and SDSC Core resources were integrated in April 2006.

creased usage of Condor to support large numbers of single-processor jobs. However, even excluding the Condor usage there was a 35% increase in the number of jobs executed in 2006.

We look at several measures, each of which produce different (but correct!) answers to the question "how many users are there?" As shown in Table 2 and Fig. 2, we differentiate between the following three groups of "users:"

Current: The group of people with user accounts.
Active: The subset of current users who have run one or more jobs (active) in a particular reporting period.
Cumulative: The group of people who have, or at one time have had, user accounts.

Differences between these measures are due to effects such as turnover in the user community and cleanup effects as allocations and machines are retired. We use snapshot numbers to track user population growth to minimize these effects on our estimate of current user population.

The growing user community reflects greater geographic distribution as well as growth in numbers during 2006, with users from 114 new institutions, active users in almost every US state and eight users from Puerto Rico.

Beyond the overall measure of the user community, however, effective planning and resource allocation requires an understanding of how the users are utilizing the facility. Traditional measures of system usage (jobs executed, bytes moved, etc.) are useful for examining individual systems, but they provide only a limited, indirect picture of the use of resources in coordinated use modalities as are becoming more and more common in distributed facilities such as TeraGrid.

In 2006 we began to collect additional data such as software use and usage of particular distributed services and we are in the process of expanding the number of "markers" we track. These "markers," combined with the experience of our user sup-

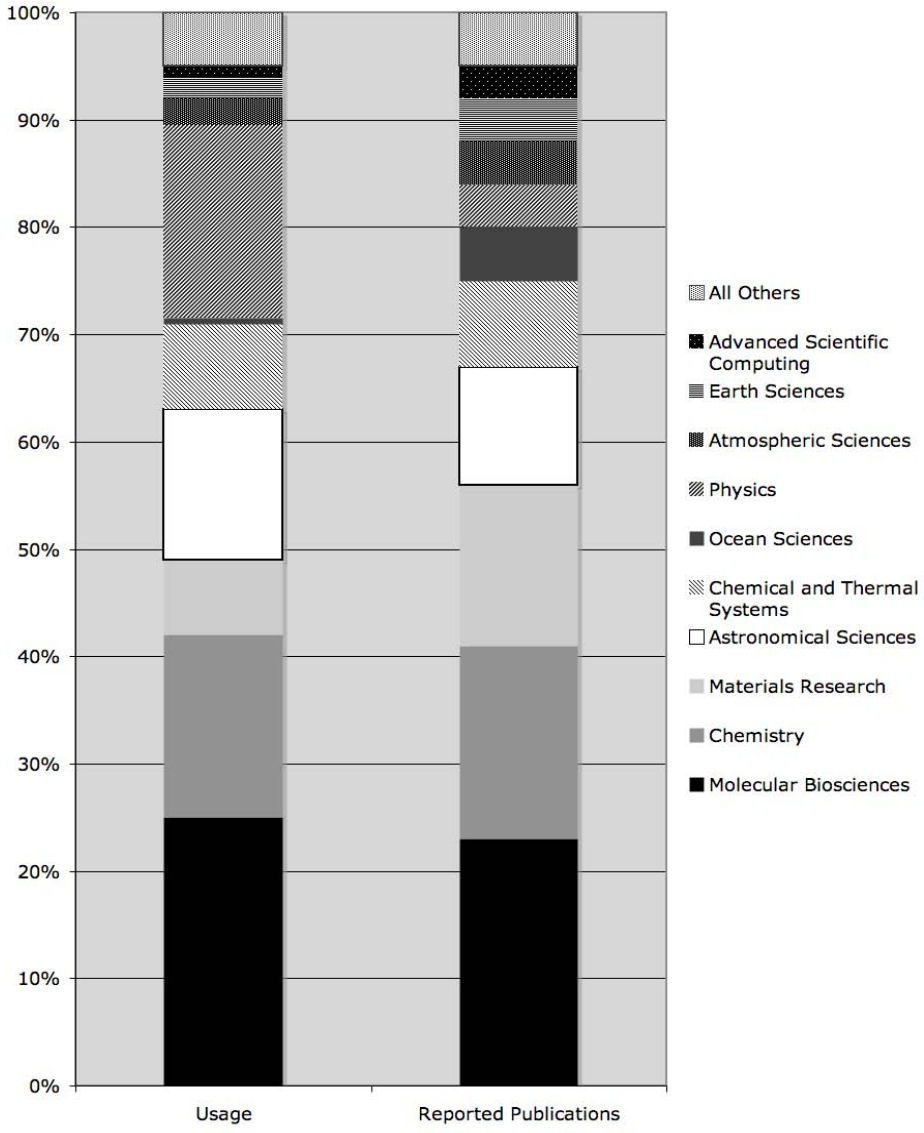


Figure 3. Publications and Usage by Science Discipline. Figure courtesy Dave Hart, SDSC.

port staff through their direct interactions with users, begin to reveal patterns of use. Shown in Fig. 4, these markers include:

- Remote Job Submission and Workflow: Remote job submission logs. GRAM [11] usage shows the trend in remote job submission (jobs and users) to five frequently used resources.
- Condor: Initial Purdue Condor flock use in 2006 was heavy and grew significantly, with a population of several dozen TeraGrid users.

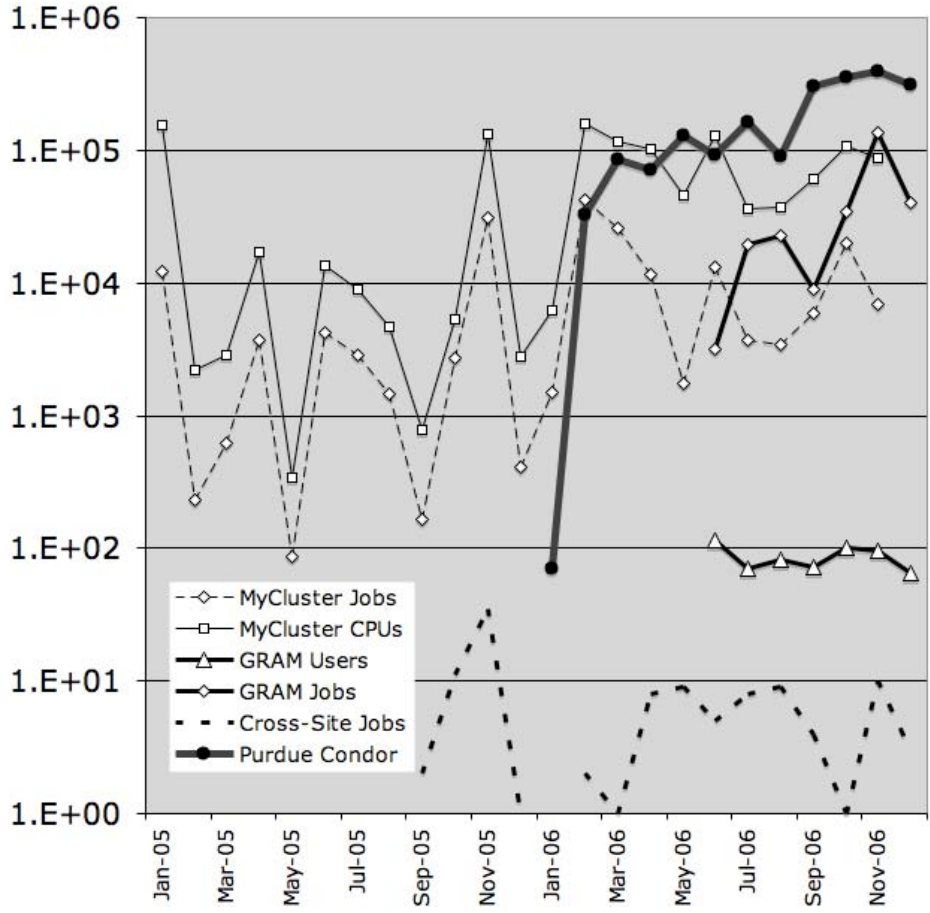


Figure 4. Initial metrics for CY2005-6 indicate growth in a sampling of distributed use modalities including MyCluster, GRAM remote job submission, and cross-site application runs. Many other modes of use are not shown in this graph, such as traditional job submission (see Table 4).

- Parameter Search: Use of the MyCluster [12] parameter search and ensemble simulation tool. MyCluster users increased from 24 to 36 this year.
- Co-Scheduling: Cross-site reservations made to our manual co-scheduling service. 15 projects ran 109 cross-site jobs. In terms of duration, 50% ran for 2–8 hours and 11% for more then 32 hours. With respect to size of jobs, 68% used 64–256 processors 15% of the jobs used more than 1024 processors.

Based on the markers above and on discussions among user services, science gateway, and technical staff members, we estimate of the number of active users (those who executed at least one job in 2006) whose typical interaction with the TeraGrid facility is best described in one of five broad categories as shown in Table 4. We note, however, that this is a very simplified overview because many teams use multiple of these modalities. For example, even those traditional HPC teams who predominantly compute in batch mode on a particular system will also use roaming allocations for opportunistic throughput enhancement.

Table 4. Estimated number of users and percent of user population for each modality of use

Use Modality	% of Active Users
Batch Computing on Individual Resources	45%
Exploratory and Application Porting	35%
Workflows, Ensemble and Parameter Sweep	10%
Science Gateway access	5%
Remote Interactive Steering and Visualization and Tightly-Coupled Distributed Computation	<5%

In 2006 there were 1,702 active users (see Table 2), roughly half of which (919) were using large or medium allocations that are primarily – but by no means exclusively – of the *specific* type. The remaining 783 users were using development allocations that are of the *roaming* type. We find that roughly half of these development users have submitted jobs to multiple TeraGrid resources, thus we characterize them as "exploratory and application porting" users. Below we describe the six different usage modalities.

1.2.1. Batch Computing on Individual Resources

The largest group of consumers of TeraGrid resources is the community of experienced HPC users who run very large jobs through batch queues. In some cases, a small research group will have a single *specific* allocation on a platform particularly suited to their work. In other cases groups will have multiple *specific* allocations to make use of a number of TeraGrid resources. These are typically large teams who are early adopters of HPC systems and adept at porting and optimizing their codes. Collaborators on these large teams may divide the simulation workload into subprojects to be executed by one designated co-PI on one designated platform. Many of these teams are beginning to explore multi-site usage scenarios through *roaming* allocations in addition to their *specific* allocations, and several of these teams use only *roaming* allocations.

1.2.2. Exploratory and Application Porting

This category represents users who have received development awards and are porting and benchmarking codes. As shown in Table 2, development award users in 2006 accounted for 47% of all TeraGrid users and 46% of those users who submitted at least one job. Roughly half of these users are interacting with multiple resources in an exploratory way, including porting and benchmarking applications to different TeraGrid resources.

While these users do not consume nearly the amount that the medium or large allocation user communities consume, they are the largest population of TeraGrid users – nearly twice the size of the medium allocation community and 70% larger than the large allocation community. Development awards are *roaming* allocations to enable users to experiment with a variety of TeraGrid systems. The base level of familiarity afforded by TeraGrid's coordinated software and services on all TeraGrid platforms makes it easier for new users to do so without having to learn the nuances of each of the dozens of individual systems. This is particularly useful in support of benchmarking codes on different TeraGrid machines, which is the traditional objective of development awards and is a required component of proposals for medium and large allocation awards.

1.2.3. Workflows, Ensemble and Parameter Sweep

Workflows are computations and data analysis tasks that are composed of a sequence of related but distinct jobs. This might include one or more data preprocessing steps, e.g. data assimilation and cleaning, followed by a series of computational steps in which one or more steps may depend upon a preceding step, followed by post analysis. Ensembles and Parameter sweeps are actually a category of workflow in which a large number of identical tasks are run logically "in parallel" followed by an ensemble analysis step. Even a job control script with more than one subtask is a type of workflow.

However, TeraGrid is ideal for much more general cases of workflows in which different tasks are executed on different resources. A workflow control process manages the orchestration of the workflow. Typical examples of these workflows use tools such as Condor's *DAGman* [13], *Kepler* [14], *Taverna* [15] and *BPEL* [16] (the standard web service workflow language.) It is not easy to distinguish jobs that are managed by one of these workflow engines, because the individual tasks look identical to other submitted jobs. However, most use either Condor or GRAM for the job submission and thus the corresponding "markers" shown in Fig. 4 are indicative of the growth of this user community. For example, we find that monthly use of remote job submission has grown from several thousand jobs in mid-2006 to several tens of thousand jobs per month by late 2006. The number of users remotely submitting jobs in this same period of time was roughly constant at 90–100, however many of these "users" are actually community allocations associated with science gateways, which use workflow tools to orchestrate complex job scenarios on behalf of the tens to hundreds of gateway users.

1.2.4. Science Gateway Usage

A typical science gateway user is interacting with a web portal, invoking applications specific to the science community supported by that gateway. Gateways provide specific application services to their user communities, executing those applications on TeraGrid platforms on behalf of those users. For security reasons, special authorization and authentication provisions apply to community user accounts, which are usually constrained to execute a fixed set of commands on the TeraGrid systems. There currently are about 25 active community allocations. For both this type of usage and the visualization usage discussed below, we are seeing demand for a different type of allocation: small but persistent awards to gain periodic access to specialized resources. As of early 2007 we are receiving reports from gateway providers with hundreds of new active users accessing TeraGrid through their community accounts, indicating rapid growth in this sector of the user community.

1.2.5. Remote Interactive Steering and Visualization and Tightly Coupled Distributed Applications

Over the past year, we have seen a growth in the need for remote interactive visualization that can only be accomplished with specialized graphics hardware. Some of these users have been traditional HPC users in the past and now have need for more sophisticated visualization capabilities; and some of these users have observational or experimental data that they are analyzing using TeraGrid resources. Users typically stage their data on the platform of interest, log into the machine, prepare the remote interac-

tive session and launch the job giving them scheduled remote interactive access to the resources. This can also be done in a variety of ways including use of the TeraGrid Visualization Gateway [17] as well as via workflows using tools such as "Portals Direct I/O" (PDIO) [18].

Another small pioneering group of users employ multiple TeraGrid resources in tightly coupled applications, comprised by two general cases: functionally decomposed simulations and data-decomposed simulations. In the first case a simulation consists of large subtasks, each of which is best suited to a specific HPC architecture. For example, model building and definition of initial conditions is done on a large-memory SMP (site A) followed by a simulation on a very large MPP (site B), and by data analysis back on the SMP (site A). In the second case, a simulation is distributed in order to harness aggregate computing or memory capacity of multiple platforms. In these cases, MPICH-G2 [19] is used to distribute a single parallel job across several Clusters, necessitating advanced capabilities such as co-scheduling.

2. Creating Sustainable Cyberinfrastructure: Software and Services Architecture

The dramatic growth of the TeraGrid user community, dominated by development allocations that are typically new HPC users, reinforces our strategy to provide a coordinated software environment across TeraGrid resources. This coordinated environment enables users to roam between machines with a single allocation, a single sign-on, and access to a common core software environment of compilers, libraries, and tools. In parallel, the science gateways initiative – with ten partner gateways in 2005 and over 20 by late 2006 – relies on a common set of interfaces, specifications, and policies that allow the gateway developers to interact in a consistent way with the heterogeneous set of TeraGrid resources.

2.1. Software Services and Capabilities

Initially the Coordinated TeraGrid Software and Services (CTSS) involved a large set of software components including the Globus Toolkit, Condor, and other capabilities. In 2006 we added capabilities to support key science use cases and to improve the robustness and scalability of existing capabilities. These enhancements were done in the context of moving toward a *Service Oriented Architecture* that relies on emerging Web Services technologies. This architecture shift also involves modularization of our software to reduce packaging, deployment, and support costs while enabling us to more rapidly respond to the software requirements of our user community. Finally, this approach introduces the potential for our users to combine TeraGrid services with other components of national cyberinfrastructure – such as campus authentication and authorization services through Internet2's Shibboleth [20] framework.

CTSS, the mechanism by which we deliver a set of common software capabilities to TeraGrid users, includes both local software packages and remote service interfaces, providing TeraGrid users with a common set of expectations for the software and services that will be present on any TeraGrid resource. We deployed the third version of CTSS in mid-2006 and are presently deploying the fourth version, CTSS 4, scheduled to complete in mid-2007.

2.1.1. New Software Capabilities in CTSS 3

The third version of CTSS was the first new version since completion of TeraGrid construction in 2004, and it represents a significant improvement over CTSS 2 with the addition of capabilities requested by our users and science gateway partners as well as process changes designed to streamline the deployment and support work of GIG and RP staff.

The key requirements that drove CTSS 3 design were as follows.

- Provide a suite of service interfaces to enable the science gateway operational model.
- Improve cross-site file transfer performance for mainstream users.
- Provide software tools to support the popular parameter sweep usage scenario.
- Provide software tools to support advanced, multi-site MPI applications.

In response to the first requirement, CTSS 3 included new service interfaces from the Globus Toolkit 4.0 (GT4) [21]. These include the WS GRAM service (for remote job submission and management), the RFT service (for managing file transfers), and the MDS4 Index Service (the basis for a TeraGrid-wide information service). These service interfaces provide high-quality mechanisms for science gateways to use TeraGrid resources via the popular Web services programming model. Usage data shows that 236,000 jobs were submitted to TeraGrid systems via the WS GRAM interface after the CTSS 3 production date in June (Fig. 4).

In response to the second requirement, CTSS 3 includes two important data movement capabilities. The first, striped GridFTP server, allows resource providers to construct multi-node data movement services that can make full use of TeraGrid's dedicated national 10–30 Gbps network for large file transfers. Usage data collected in 2006 shows that these GridFTP servers moved, on average, between 0.6 and 1.6 TB of data into and out of TeraGrid systems every day. To make this capability more accessible to new users, CTSS 3 includes a second capability: tgcp (TeraGrid copy). Tgcp presents a familiar Unix secure copy (scp) syntax to perform serial, striped, and reliable (RFT) GridFTP file transfers. Tgcp also uses knowledge about TeraGrid's GridFTP server configurations to automatically apply tuning parameters that optimize file transfer performance and to select appropriate service endpoints. With this capability, users with no little or no knowledge of the specific systems or network configurations (or GridFTP) can benefit from tuned, high-performance file transfer services.

In response to the third requirement, CTSS 3 included MyCluster [22], developed at TACC. MyCluster works in conjunction with Condor to simplify the execution of large-scale parameter sweep applications. CPUs on multiple TeraGrid systems are allocated using standard job submission interfaces and then used to execute parallel tasks. Figure 4 shows the growth in MyCluster usage during 2006.

The fourth requirement arose from the fact that a small number of early adopter science teams have found that they can now solve previously intractable problems by executing their simulation codes across multiple TeraGrid resources, optimizing their algorithms to compensate for wide-area communication latency. CTSS 3 includes an updated, more robust, and better-documented deployment of MPICH-G2, a tool that supports both local and wide-area execution of applications built using the standard MPI programming model. Several user teams have used this deployment of MPICH-

G2 for award-winning simulation runs that are on the leading edge of scientific simulation capabilities.

2.1.2. Use of TeraGrid's Integrated Software

Understanding usage and usage trends is essential to identify and prioritize improvements. Most TeraGrid service interfaces (e.g., ssh, GridFTP) record usage information in local log files, though the kinds of information tracked vary from service to service. This log data can be collected from multiple systems to produce usage reports. However, several services (from the Globus Toolkit in particular) are instrumented to report usage to a central *listener* service, where data is stored in a database.

In addition to the software and capability use data we described earlier, we note the following adoption indicators:

- **Single sign-on**. In the second half of 2006 there were 597 unique users who used TeraGrid's MyProxy server, which enables access through a single login (credential) to all TeraGrid systems.
- **Remote Job Submission**. TeraGrid GRAM interfaces (see Fig. 4) were used by at least 50 individual and community accounts across TeraGrid, supporting at least 526,000 job submissions across TeraGrid sites from June through December. The GRAM interface supports science gateway integration (e.g., NanoHUB, BioPortal), large-scale parameter studies (e.g., MyCluster users), and large-scale science workflows (e.g., SCEC, GADU).
- **Striped GridFTP**. TeraGrid GridFTP servers moved between 0.6 and 1.6 TB of data per day on average during 2006 into and out of TeraGrid systems. Data movement illustrates the use of TeraGrid systems as elements of the end-to-end scientific workflow.

2.1.3. New Capabilities Developed for CTSS 4

The primary driver for CTSS 4 is to reduce deployment and support effort required by RP sites while increasing the control an RP site has regarding the services offered through TeraGrid. By improving the capability delivery process we also enable improved stability and reduced cost for RPs. By modularizing CTSS we allow for more agile deployment of new capabilities – such as upgrading or adding capabilities without changing the entire CTSS deployment.

We will complete the deploy CTSS 4 in mid-2007, introducing significant improvement in our ability to deliver new capabilities to the user community by decentralizing the integration process that had been used for CTSS 1–3. Our new process allows other software providers (and RPs) to add capabilities to CTSS independently. CTSS 4 deployment and enhancement will employ a formal, open change management process for proposing and deploying new CTSS capabilities.

We are deploying a number of new capabilities with CTSS 4, many of which are driven by the requirements of science gateways and the TeraGrid user portal. These include:

- Queue prediction using the Batch Queue Predictor service from the Network Weather Services team [23] – This will allow users to compare the expected queuing delay of a given job on various TeraGrid systems.

- Science Gateway Audit Interface – This allows science gateway providers to track the use of their community allocations based on usage by local (gateway) users.
- Integrated Information Service – This enables automatic service registration, service discovery interfaces, and automated mechanisms for updating documentation.
- CTSS 4 Science Workflow Support kit – This provides documentation and tools to support this style of scientific operation, including mechanisms to make it easier to add applications into workflow models.

2.2. Integrated Software Environment Management

2.2.1. CTSS 4 Design and Costs

Deployment of CTSS involves cooperative effort between GIG and RP staff. The GIG produces software packages for RPs that include pre-built software for individual TeraGrid systems with customized deployment, configuration, and testing instructions. RPs deploy these packages on their HPC resources and manage operational issues. GIG coordinates operational issues, including enhancing and operating Inca (our validation and verification suite [24]) and coordinating service outage review and response. The Inca system provides monitoring of CTSS and was upgraded in 2006, adding features to better identify, analyze, and troubleshoot user-level Grid failures, thereby improving TeraGrid stability. The GIG also coordinates interactions between RPs and software vendors to ensure that problems and fixes identified by resource or software providers were made available to the entire TeraGrid community.

Overall, RPs estimate that maintaining CTSS on their systems requires an additional 0.25 to 1.75 FTEs beyond what they would otherwise spend on software maintenance. The large variation is driven by two dominant factors: (1) how much of CTSS would be deployed if the RP were not a TeraGrid partner, and (2) how able the RP is to take advantage of the GIG assistance. The first factor is largely determined by the degree to which the RP's resources are used by other cyberinfrastructure initiatives. The second factor is determined by the uniqueness of the RP's resources and practices. There is a wide range of diversity on both points.

2.2.2. CTSS Design and Structure

The first step to moving to a modular CTSS 4 and a service oriented architecture was to restructure the CTSS 3 software into a series of capability kits, each focusing on a specific set of related user capabilities. Examples include: remote job submission, remote login, and science workflow support. A single "TeraGrid Core Integration kit" was designed to provide the management capabilities that integrate any TeraGrid resource with the rest of the TeraGrid facility. (These include common security mechanisms, capability registration mechanisms, verification & validation mechanisms, and capability deployment and instrumentation mechanisms.) We also documented the CTSS design and delivery process so that anyone in the TeraGrid community can now define and deliver new CTSS capabilities.

Beginning with CTSS 4, CTSS capabilities will be managed in a distributed fashion by teams of experts in particular capability areas drawn from the RPs and GIG, and potentially external software partners. New capabilities can be deployed on independ-

ent schedules. The Software Working Group and GIG operations staff will continue to coordinate new capability deployment schedules throughout the TeraGrid community.

2.2.3. Software Build and Test

The NMI [25] Software Build-Test mechanisms play a key role in our software management process. A significant subset of CTSS 4 capabilities are prepared using these build-test mechanisms, which will be deployed on all TeraGrid resources. Our partnership with the Virtual Data Toolkit (VDT) team at the University of Wisconsin ensures that software prepared for CTSS and software prepared for VDT (used by other Grid projects such as OSG and EGEE) use the same versions, patches, and builds for common platforms.

2.3. Authentication, Authorization, and Accounting

TeraGrid supports two authorization methods: a) users are individually registered with TeraGrid and associated with a particular project, or b) users register with a Gateway that acts as a proxy to invoke TeraGrid services through a community account. Current work is focused on introducing more broadly the virtual organization approach of (b) while retaining the accountability benefits of (a). We have already put in place a TeraGrid Kerberos [26] Realm to support the User Portal authentication and are leveraging this to support single sign-on functions across the TeraGrid.

U.S. campuses, home for most of our users, are creating robust and interoperable Identity Management systems. Most notable among these is the *inCommon* Shibboleth federation. Working with Internet2 partners, we are deploying a testbed in 2007 for using campus credentials to authenticate to the TeraGrid. An evaluation will be held in June 2007 on whether we should proceed to an initial production deployment currently targeted for early 2008. We are working with the community to establish a set of guidelines and agreements that can readily be re-used by new participants.

To support a national allocations system that allows users to have either specific or "roaming" allocations, as outlined earlier, we employ a distributed resource accounting and accounts management system. Originally developed at NCSA, this system uses a central user and usage database and a standard messaging system to exchange accounting information as well as requests such as to create accounts or map user credentials to accounts. Each TeraGrid resource reports usage to the central database, which tracks balances on allocations and supports reporting and queries from resource providers as well as from the users via the user portal.

Load in our distributed accounting system grew by more than a factor of three in 2006. This was fueled by growth factors outlined in §1.1: seven additional HPC systems, 2500 new users, more than double the usage. Changes in use modalities also have a non-linear effect on the system. For example, parameter sweep studies supported by Condor flocks or MyCluster software can produce thousands of usage records where on a traditional supercomputer they would produce a single record.

We are exploring strategies to influence overall TeraGrid workload to exploit lower utilization on some RP systems. Given the large variety of system sizes and types within TeraGrid, we find that there is opportunity- and need- to optimize load across the system in order to provide improved service for end users. Our distributed accounting system and policies allow for non-uniform billing where an RP would set a charging rate to either promote increased – or decreased – usage based on load, queue

length, etc. A simple strategy we are exploring in 2007 is a lower rate for *roaming*, or opportunistic resource use, on lower utilized systems.

We are also investigating "on-demand" computing services that become more practical with a large set of resources than for a single center. We have developed a system called Special Priority and Urgent Computing Environment (SPRUCE [27]) that supports priority "tokens" that users can use to flag a job as "urgent," with several levels of priority. Resource providers determine local policy for responding to urgent computing requests. For example, one resource provider may elect to suspend all running jobs in order to immediately run a high-priority urgent job while another may allow the job to go to the "front of the queue" for "next-run" status. Initial implementation of SPRUCE at the University of Chicago and Argonne National Laboratory is exploring various policies including the notion of offering a cluster as a "pre-emptible" service for a lower "price" because the user expects that his or her job may be suspended in the event that an urgent job arrives.

3. Engaging the Community

TeraGrid's user community is not only growing in size but is also diversifying with new programs such as Science Gateways. Thus our user and community engagement strategies attempt to map to the needs of various types of users. Below we describe traditional user support strategies, early-adopter support strategies, and our Science Gateway program that effectively uses a "wholesale-retail" model. In such a model, TeraGrid support is focused on gateway service providers who in turn are engaging entire communities of users.

3.1. User Support Strategies

User support within a computing center is well understood, but differs significantly from user support in a distributed infrastructure. Beyond the complexity of a distributed system from a problem diagnosis and resolution perspective are the multi-organizational dynamics of coordinating the efforts of staff. A key focus of our coordination strategy in this area has been to develop a rich set of interactions and interconnections among support staff at multiple centers, harnessing the unique strengths each participating group brings to bear.

Our user support approach involves four integrated programs that are coordinated by the GIG and staffed from both GIG (3 FTE) and the RPs (20 FTE):

- ***Proactive*** User Training and Support
- ***Persistent*** Online Tools and Resources: Website, Knowledgebase, User Portal
- ***Responsive*** TeraGrid Operations Center (TOC) Helpdesk
- ***Advanced*** TeraGrid Applications Support (ASTA).

In addition to the 23 FTE dedicated to user support, the full user services team effectively involves over 80 staff members (many of whom are not funded by TeraGrid) from throughout the project who monitor the user services and trouble ticket distribution mailing list and participate in problem resolution.

Our user support strategy is driven by several user support requirement patterns. Table 5 summarizes the top ten issues that largely characterize the work of the user services team. The table also helps to illustrate how the team addresses various types of

Table 5. Top Ten User Support Issues/Questions

Locally Handled	Coordinated	Centrally Handled
Job turnaround (wait)	Job submission	Account balances
Job failures	Data transfers (WAN)	Access/outages
Code porting & optimization		Logins/passwords
Third-party packages		
File system problems		

issues. Some items are handled directly by the central helpdesk at the TeraGrid Operations Center (TOC), others are assigned to the relevant RP site support team, and some require cooperative diagnosis among different RP sites.

3.1.1. User Training and Support and Helpdesk

To ensure that new users are able to smoothly begin working, we assign a member of the user support team to each new PI with either a large or medium allocation award. This "ombudsman" will contact the user and will provide ongoing personal support. As was discussed in §1.1, we have seen a dramatic increase in new users through development awards, such that these users now make up roughly half of the user community. For these users, we provide startup instruction materials and TeraGrid help desk contact information with their "welcome packet" of materials. Much of our work in online tools and resources, including the user portal, is aimed at ensuring that we have a robust support structure in place for this most rapidly growing portion of the community.

3.1.2. Online Tools and Resources: Website, Knowledgebase, User Portal

TeraGrid online resources include the TeraGrid main website, TeraGrid User Portal, and a TeraGrid Community Wiki. The main website is the primary access point today for documentation, knowledgebase, and for accessing the allocations proposal system. A content management system allows staff from the RP sites to update site-specific information regarding resources and services. The TeraGrid wiki is the primary collaboration site for project planning, internal policy development, internal and public reports, and other activities. The TeraGrid makes use of a Knowledge Base (based on Knowledge Base Technology from Indiana University) to provide a convenient interface for users to search and find solutions to technical problems.

The common issues experienced by the support team as shown in Table 5 also drive the development of specific user tools and the evolution of the user portal. For example, several of the "top ten" issues were among the first to be addressed in the initial launch of the user portal. Users can manage logins and passwords, check allocation account balances, and use prediction tools and resource queue status information to optimize job turnaround.

The TeraGrid User Portal, launched in May 2006, provides a single point of entry for TeraGrid users to all TeraGrid processes, including cross-referenced access to the website, Wiki, and other online resources. The initial version included basic tools for users to manage allocations: allocation usage monitoring, system account directory, resource system monitoring, and user documentation. During its first 6 months of operation, more than 20% of TeraGrid users had authenticated and used the User Portal. Among the top 20 most active user portal users, six were from large allocation projects,

demonstrating that even experienced users are finding value in this gateway to TeraGrid.

We expect continued adoption of the User Portal in 2007 as we expand the available tools such as batch queue and data bandwidth prediction interfaces, and as we provide richer allocation management tools. We will also introduce the concept of science domain views to provide portal users customized default interfaces depending on the domain of science views they select.

3.1.3. Advanced Support for TeraGrid Applications (ASTA)

The Advanced Support for TeraGrid Applications (ASTA) Program associates application consultants with specific user teams, on average involving 25% of a consultant's time for 6–12 months. Each project involves a detailed scope of work that generally focuses on application software development necessary to maximize the effectiveness of the use of TeraGrid resources and capabilities toward the user's scientific goals. ASTA projects are typically attempting to harness advanced, often new, TeraGrid capabilities for which there is not yet sufficient operational experience to optimize our general support and documentation. Thus, ASTA projects help to "debug" these advanced features, including assisting us in determining their utility.

Due to the growth in demand for the ASTA program we have developed a policy to introduce peer review, whereby ASTA support can be requested through the national allocations process using the same mechanisms that grant TeraGrid allocation awards.

We have planned a new ASTA selection model that will employ the xRAC committee to review requests. Among our strategic 2007 ASTA goals are to promote:

a) Scaling to Petascale and ~10^4 cores,
b) Complex workflow development embedding Petascale applications assisted via (a) into the entirety of TG infrastructure, and
c) Diversity of disciplines and modalities of resource usage.

3.2. TeraGrid Science Gateways Initiative

The Science Gateways program promotes and supports the use of HPC resources through community-designed interfaces, recognizing that many of today's scientists routinely use desktop computing applications and web browsers to conduct their work, including utilizing remote HPC resources. The gateway program began with eight specific prototypes spanning seven disciplines, each of which had external funding to build a community-specific infrastructure and an existing user community. This included seven web portals and a community Grid (Open Science Grid).

The gateway model to access resources is available to any academic developer, and this opportunity has been highlighted at the TeraGrid website as well as widely advertised through workshops and outreach events. TeraGrid support staff work directly with developers who are providing capabilities for their communities in the same fashion that our user services programs (see §3.1) assist individual HPC users.

The gateway architecture defines programming interfaces and web services that developers can use to bring resources of the TeraGrid to a community of users within the environment – generally a web portal – with which they are already familiar. This effectively adds HPC resources to the scientific, and education portfolio of these communities without introducing a steep learning curve. The adoption of this model has been rapidly growing, increasing the importance of our providing the necessary func-

tionality and documentation to enable developers to incorporate TeraGrid resources into their community infrastructure. The gateway program is supported by GIG-funded staff who focus on specific technologies required by multiple gateways, such as job audit and on-demand computing support. Initially (in 2005–6) we dedicated GIG staff members to the eight original gateways in order to focus on developing and building a scalable set of processes and policies to start the program. During 2007, the GIG-funded staff will transition from focusing on the initial gateway prototypes to forming an integration team that can assist new gateways, selected through a peer-review process similar to that used with the ASTA program. Already during 2006 this team has expanded their support to over 20 gateway partners.

Gateway work began in 2005 with a survey of 10 projects from a variety of disciplines and with a variety of access models, and the results of this study drove our initial set of priorities for gateway work as well as requirements for GIG software integration work (e.g. the requirement for web services, see §2.1). While gateway developer needs are often distinct from scientists using HPC systems at the command line, we found a significant core of common service and capability requirements shared by *Deep* HPC users and *Wide* gateway users and developers. By working with some pathfinders, we have been able to deploy needed capabilities, while educating developers on changes they need to make to operate in a shared, production environment. As we develop methods and processes for gateway integration, we implement these and document them via an online primer, so that integration is easier both for subsequent gateways and for subsequent resource providers.

3.3. Training, Outreach and Community Engagement

TeraGrid training, education, and public outreach programs leverage the efforts of RP sites in a coordinated fashion to allow for common planning, optimal event scheduling, and sharing of expertise and materials. In 2006 alone we supported over 100 training, education, and public outreach events, reaching thousands of educators and students in hundreds of secondary, undergraduate, and graduate institutions. Evaluation of these programs is extensive in terms of both internal and external (a separate NSF award to the University of Michigan) methods ranging from focus groups to surveys and interviews.

We organize our education, outreach, and training programs with a comprehensive approach we refer to as "HPC University," where integrated scheduling and event information allow a user or prospective user to develop a personalized training plan drawing from tutorials, workshops, and other events across TeraGrid as well as in the broader international and US community. This concept involves:

- A regular series of training sessions conducted by TeraGrid RP and GIG staff to address a variety of topics from introductory to advanced topics (new user startup, user portal, parallel programming, data management, data analysis, visualization, grid computing, etc.).
- Coordinated summer institutes and workshops at RP sites to introduce users to TeraGrid resources. Training will also be provided to TeraGrid staff to ensure they are fully up-to-date on the latest tools, technologies, and methods.
- Curriculum development, with the support from the SC07-09 [28] Education Programs, working with undergraduate faculty and high school teachers on

integrating computational science, scientific computing and grid computing resources, tools and methods into the curriculum.
- Student internships at Resource Provider and GIG sites.

4. Implementation: Organization, Structure, and Governance

TeraGrid is a facility involving resources and services that are provided by multiple, autonomous, institutions. Resource providers (RPs) are independently funded, by the National Science Foundation and other sources, to deliver sophisticated portfolios of HPC resources, support, and related services. The Grid Infrastructure Group (GIG), funded through a separate grant to the University of Chicago, is a distributed team charged with a variety of integrative functions. GIG leadership and management roles are filled with individuals selected from across the project based on expertise and merit. Similarly, the majority of GIG staffing is drawn from partner institutions via subawards to partner sites, each with a detailed statement of work identifying the tasks and responsibilities of the individuals at that site who are funded as part of the GIG.

The distributed GIG function provides for several key TeraGrid-wide capabilities and services, predominantly through subawards to experts at RP sites, including:

- A TeraGrid operations center and helpdesk,
- Common services such as authentication and authorization services, information services, a user portal, and various internal and external websites,
- Integration of new capabilities (software, policy, interfaces) to address user requirements,
- A nation-wide, dedicated optical network backbone and hubs to interconnect the RP sites, each of whom is responsible for maintaining a connection to a hub on the backbone network (hubs are located in Los Angeles, Denver, and Chicago).
- Common processes such as accounting, authorization, and allocations peer review.

In addition, the GIG provides coordination for a larger set of activities that involve both GIG and RP staff, including:

- User support functions and programs such as ASTA,
- Education, Outreach, and Training initiatives,
- Overall software and service operation and coordination,
- Planning and prioritization of strategies and architecture,
- Organizational structures for coordination, policy, and governance.

4.1.1. Technical Strategies and Planning

Because the GIG function of TeraGrid is inherently distributed – at both the management and staffing levels – it provides an excellent platform for coordination and facilitation for project-wide planning. As with most institutions and projects we have a set of "working groups" that are focused on technical or service areas, involve staff from each participating institution, and are effective at ongoing coordination of services. However, working groups can readily become technical "stovepipes" and because they are responsible for operational services they tend to minimize change in order to opti-

mize for stability. As priorities and requirements change, however, it is necessary to explore changes and new approaches as well as to bring together individuals from different technical areas.

We created a structure called a "Requirement Analysis Team" (RAT) that addresses these two issues. A RAT is created for a finite period of time – generally 8–10 weeks – in order to explore a particular opportunity or challenge that cuts across multiple technical and policy realms. A RAT operates based on a written charter and the responsibility of the RAT is to create a set of recommendations. There have been nearly 20 RATs in the past two years of the TeraGrid project, each of which has produced a detailed set of analysis and recommendations that inform and in most cases define TeraGrid strategy and policy.

4.1.2. Governance: The TeraGrid Forum

The decision-making process in the TeraGrid project includes both local and collective functions. As each resource provider, and the GIG, are independent entities there is no single, top-down, authority that dictates policy and technology. Cooperative decision-making is accomplished through the TeraGrid Forum, a body including one representative from each participating institution. In March 2006 the Forum "ratified" a consensus-based democracy decision-making process where major policies are recorded, along with consensus records, in a persistent document series.

5. Conclusions

TeraGrid began as a cooperative project of four institutions and 6 resources in late 2001 and has grown to over a dozen institutions with over 20 major HPC resources. Since completing construction in 2004 TeraGrid's user community has grown from several hundred to several thousand users, including many users who are new to the HPC environment.

As with many "grid" projects, the bulk of early use has been traditional, rather than distributed, usage modalities. However, during 2006 the TeraGrid project saw dramatic adoption of a number of new use modalities including workflow, remote job submission, and parameter sweep. During the same period of time a new generation of HPC user emerged comprised of two types of new users, each with unique expectations. The first are researchers whose base expectation is that they can access any of the TeraGrid systems rather than being tied to a particular computer at a particular computing center. The second are users who access HPC services through web portals and via web services technologies.

We believe that one of the keys to continued success at providing for the needs of these different, and growing, user communities is the coordinated provision of standard services on HPC systems, recognizing that they are not stand-alone capabilities to which users must adapt their work. Rather, they are rather building blocks users wish to incorporate into their own science environments. Critical to this approach is the need to actively engage the scientific user community in order to understand their requirements, deploying services aimed at meeting the scientific objectives of the user community.

Acknowledgments

The TeraGrid project has been supported through a variety of funding and in-kind contributions in addition to multiple grants from the National Science Foundation. State support has come from the states of California, Illinois, Indiana, Pennsylvania, and Texas. Institutional support has come from Carnegie Melon University, Indiana University, Purdue University, University of California-San Diego, University of Chicago, University of Illinois at Urbana-Champaign, University of Pittsburgh, the University of North Carolina, California Institute of Technology, and the University of Texas. Corporate support has come from Cray, Dell, IBM, Lilly Endowment, Qwest Communications, and Sun Microsystems. Several hundred staff members from partner institutions contribute to the TeraGrid facility.

References

[1] National Science Foundation: www.nsf.gov.
[2] "Guidelines for Planning and Managing the Major Research Equipment and Facilities Construction (MREFC) Account," www.nsf.gov/bfa.
[3] The NSF High Performance Computing System Acquisition: Towards a Petascale Computing Environment for Science and Engineering (NSF 05-625) has proposal deadlines of 2/06, 11/06, 11/07, and 11/08 for acquisition of HPC systems to be integrated as part of TeraGrid, with anticipated $30M available annually. The "Track 2" solicitation is available at www.nsf.gov/pubs/2005/nsf05625/nsf05625.htm.
[4] Catlett, C., Beckman, P., Skow, D., and Foster, I. "Creating and Operating National-Scale Cyberinfrastructure Services," CTWatch Quarterly, Volume 2, Number 2, May 2006. www.ctwatch.org/quarterly/articles/2006/05/creating-and-operating-national-scale-cyberinfrastructure-services.
[5] Phil Andrews, Bryan Banister, Patricia Kovatch, Chris Jordan, Roger Haskin, "Scaling a Global File System to the Greatest Possible Extent, Performance, Capacity, and Number of Users," Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05), 2005.
[6] Hedges, R., Loewe, B., McLarty, T., and Morrone, C. 2005. Parallel File System Testing for the Lunatic Fringe: The Care and Feeding of Restless I/O Power Users. In Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (Msst'05) – Volume 00 (April 11–14, 2005). MSST. IEEE Computer Society, Washington, DC, 3-17. DOI= dx.doi.org/10.1109/MSST.2005.22
[7] Data Collections requirement analysis team report, December 2006, Kelly Gaither et al. www.teragridforum.org/mediawiki.
[8] High-Performance Linpack benchmark (HPL) is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers. It can thus be regarded as a portable as well as freely available implementation of the High Performance Computing Linpack Benchmark. From Dongarra et al., University of Tennessee, www.netlib.org/benchmark/hpl.
[9] The Top500 project, initiated in 1993, is a collaboration between the University of Tennessee, the University of Mannheim, and the U.S. Department of Energy Office of Science National Energy Research Scientific Computing Center (NERSC) at the Lawrence Berkeley National Laboratory (LLNL). See www.top500.org.
[10] "Allocations and Usage Requirements Analysis Team" (AURA) report, October 2006, Richard Moore et al. www.teragridforum.org/mediawiki.
[11] Globus Resource Allocation Manager (GRAM), www.globus.org/toolkit.
[12] MyCluster: www.teragrid.org/userinfo/jobs/gridshell.php.
[13] Directed Acyclic Graph Manager (DAGman): www.cs.wisc.edu/condor/dagman.
[14] Kepler is an open source, cross-platform workflow software project: kepler-project.org.
[15] Taverna development is led by Tom Oinn at the European Bioinformatics Institute with the main development team at the University of Manchester and funded through the Open Middleware Infrastructure Institute UK (OMII-UK) myGrid node: taverna.sourceforge.net.
[16] Business Process Execution Language (BPEL): en.wikipedia.org/wiki/Business_Process_Execution_Language.

[17] Binns, J., DiCarlo, J., Insley, J. A., Leggett, T., Lueninghoener, C., Navarro, J., and Papka, M. E. 2007. Enabling community access to TeraGrid visualization resources: Research Articles. Concurr. Comput.: Pract. Exper. 19, 6 (Apr. 2007), 783-794. DOI= dx.doi.org/10.1002/cpe.v19:6.
[18] Stone, N. T. B., Balog, D., Gill, B., Johanson, B., Marsteller, J., Nowoczynski, P., Porter, D., Reddy, R., Scott, J.R., Simmel, D., Sommerfield, J., Vargo, K., and Vizino, C. "PDIO: High-Performance Remote File I/O for Portals-Enabled Compute Nodes," Proceedings of the 2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06/ ISBN#:1-932415-89-0/CSREA), Editor: Hamid R. Arabnia, pp. 925-930, Las Vegas, USA, 2006.
[19] MPICH-G2, Nick Karonis, Northern Illinois University: www.niu.edu/mpi.
[20] Internet2 Shibboleth Framework: shibboleth.internet2.edu.
[21] WS GRAM, RFT, and MDS4: www.globus.org/toolkit.
[22] MyCluster: www.teragrid.org/userinfo/jobs/gridshell.php.
[23] Network Weather Service project, Rich Wolski et al., University of California at Santa Barbara: nws.cs.ucsb.edu.
[24] S. Smallen, K. Ericson, J. Hayes, C. Olschanowsky, "User-level Grid Monitoring with Inca 2", to appear in the Proceedings of the HPDC 2007 Workshop on Grid Monitoring, June 2007.
[25] NSF Middleware Initiative: archive.nsf-middleware.org.
[26] Kerberos: web.mit.edu/Kerberos.
[27] SPRUCE, Beckman et al., Argonne National Laboratory: spruce.teragrid.org.
[28] SCxy, The International Conference for High Performance Computing, Networking, Storage, and Analysis: www.supercomp.org.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Applying the Provenance Data Model to a Bioinformatics Case

Paul GROTH [a] and Steve MUNROE [a] and Simon MILES [b] and Luc MOREAU [a,1]
[a] *School of Electronics and Computing Science, University of Southampton, UK*
[b] *Department of Computer Science, King's College London, UK*

Abstract. Scientists and, more generally end users of computer systems, need to be able to trust the data they use. Understanding the origin or *provenance* of data can provide this trust. Attempts have been made to develop systems for recording provenance, however, most are not generic and cannot be applied in a general manner across different systems and different technologies. Moreover, many existing systems confuse the concept of provenance with its representation. In this article, we discuss an open, technology neutral model for provenance. The model can be applied across many different systems and makes the important distinction between provenance and the way it can be generated from a concrete representation of process. The model is described and applied to a grid-based example bioinformatics application.

Keywords. Provenance, data model, bioinformatics.

Introduction

With the ever increasing proliferation and complexity of computational systems that underpin so much of human activity, people are increasingly becoming dependent on the data that these systems produce. The amount of data produced is growing exponentially, and ensuring that it can be trusted is becoming ever more important as increasing numbers of decisions we make rely so heavily upon it. Consequently, methods are required that can provide the necessary guarantees that the data they produce can be trusted, validated and replicated. In the context of Grid and High Performance Computing, the amount of data produced can be overwhelming and, in order to aid users of such systems better understand and trust their data, techniques are required that can provide a historical account of how such data is produced. The history, lineage or *provenance* of a given piece of data provides understanding of how it was that the data came to be as it is. This understanding enables users to validate data by providing the means to examine the processes that produced it for fitness for purpose, compliance to regulations, replication, validation and examination.

Recent work conducted at the University of Southampton under the PASOA and EU Provenance projects has addressed the problem of determining the provenance of data for open and heterogeneous applications. Work has been conducted to define an architec-

[1]Corresponding Author: Luc Moreau, L.Moreau@ecs.soton.ac.uk

tural framework that developers can use to enable the capture of information about processes within their applications enabling them to answer questions about the provenance of data. The work defines a technology neutral framework for representing, capturing, and querying the provenance of data, and an *open data model* that provides the means for developers to collect and organise provenance-related information. The combination of the architecture and the open data model enables provenance capability to be incorporated into a wide range of applications for a variety of application domains. For example, the architecture and data model have been applied to the health industry in an organ transplant management application [2], design engineering in an aerospace engineering simulator [12], medicine via a fMRI brain atlas imaging example [2] and bioinformatics in the context of a compressibility application [10].

Many existing papers exist that describe various aspects of the work by both the PASOA and EU Provenance projects [3], however, in this article we focus on describing how the open data model can be applied to an updated version of the above cited bioinformatics example. By presenting such a description, we aim to provide an account of the data model use that will aid other developers adopt it for their own applications. This paper makes two contributions:

1. an explanation of our open data model tailored to developers and;
2. an evaluation of the usage of the open data model in a grid-based bioinformatics experiment.

The article proceeds as follows: In the next section, we provide a high level overview of the open data model. In Section 2, we introduce the bioinformatics example. Section 3 describes in detail how the open data model was applied to the bioinformatics example. In the context of this example, we then discuss the performance of our approach in Section 4. In Section 5 we discuss related work, and in Section 6 we offer some concluding remarks.

1. A Conceptual Model For Provenance

Provenance, understood as it is in the world of Art, refers to the documented history of a given object such as a painting — who painted it, who owned it, who restored it, what restoration work has been done to it, where it has been kept and so on. This documented history enables scholars, owners and viewers to both better appreciate the significance of the object and to have confidence of its authenticity. The importance of provenance in the context of Art scholarship and appreciation is mirrored in the world of computer applications and data. Applications produce, consume and operate on data, and to ensure that we can trust applications it is necessary to be able to inspect their data, i.e. its origins in the processes that produced it. Having this information enables trust in the data and also provides other benefits besides, such as the ability to reproduce the data, understand why data is not as expected and prove that the creation of the data complies with stated policies and meets assumptions.

The start of our conceptual model for provenance begins with our definition of provenance for computational systems, which is *the provenance of a piece of data is the*

[2] http://twiki.ipaw.info/bin/view/Challenge/WebHome
[3] http://www.gridprovenance.org/biblioPages/

process that produced that piece of data. A unique aspect of our model that differentiates it from other approaches (See Section 5 for a review of related work) is the distinction we make about the information we use to obtain provenance and provenance itself. The former relates to information we gather about the process that produced the data — termed *process documentation* — and the latter refers to the results returned from performing a *query* over process documentation. In other words, the provenance of a given data item is what is returned when a query is performed over the documentation of the process that produced it. This distinction is vital because it allows the results returned from provenance queries to evolve and become more comprehensive as more documentation is created.

1.1. A Concept Map for Provenance

In order to provide an overview of our model for provenance, Figure 1 shows a concept map inspired from our previous work on specifying an open provenance architecture [9]. The concept map reflects the above mentioned distinction between provenance and its representation in a computer system.

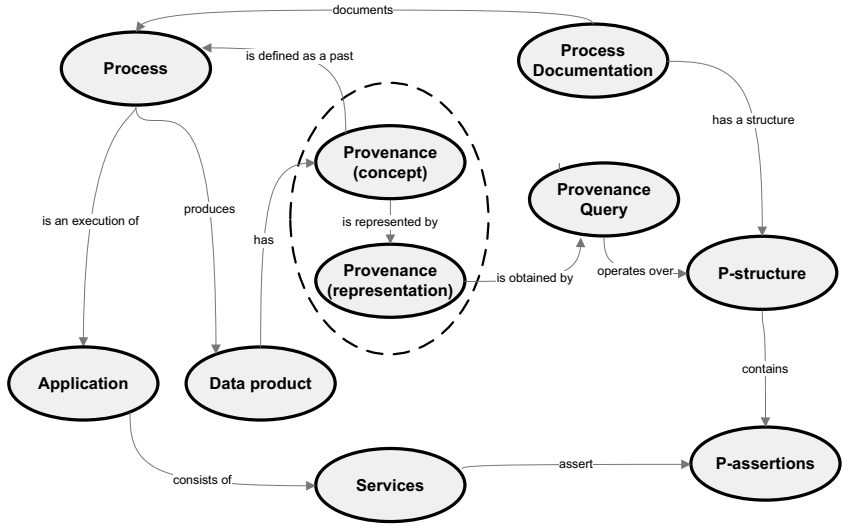


Figure 1. Concept Map for Provenance

A user who cares about the provenance of a data product has an interest in specific kind of information related to the process that led to the data product (cf. restoration vs ownership for a work of art); likewise, their needs can identify how far in the past the information should come from, hence the aforementioned representation of provenance as the result of a provenance query operating over the documentation of a process. Such documentation has a concrete representation, which we refer to as the *p-structure*. It consists of a structured set of *p-assertions* made by the different components of the application, whose execution produced the data product we consider the provenance of. Such p-assertions are asserted by software components and describe the components'

involvement in a process. If all software components produce a description of their execution, process documentation would be very complete, and from it, many useful questions about the application could be answered through provenance questions. Within our model for provenance, such generated process documentation is stored in special repositories called *provenance stores*. These stores provide standard interfaces to allow software components to record and query the stored process documentation.

We note that, while our discussion focused on electronic data and computer-based applications, nothing in this concept map restricts us to the digital world. In fact, the provenance of a physical artefact could also be obtained by querying the documentation of the physical process that led to this artefact.

2. A Bioinformatics Use Case: The Amino Acid Compressibility Experiment

With the above overview, we are now in a position to show how our model represents data in a concrete domain application. Specifically, we will explain how the representation of provenance, i.e. process documentation, is structured and how this maps onto the example application. Note, that in this article we do not consider the techniques used to map our model of provenance onto an application, i.e. what must be done in order to make an application capable of generating process documentation. For this, we have specified the PrIMe methodology, a full description of which can be found in [15].

Briefly, the methodology consists of three phases. In Phase 1, use cases for provenance within the application are identified. From these use cases, important information items are identified whose provenance may need to be determined after application execution. Phase 2 maps the application into the interaction (i.e. Service Oriented Architecture) based perspective of the open data model and identifies which components within the application need to be modified to assert process documentation. Finally, in Phase 3, the application is modified to generate and record process documentation. While we have followed this methodology when adapting this use case, the aim of this paper is not to discuss the use of the methodology; instead, it aims to describe how the application's runtime information can be represented by our model after it has been made provenance-aware.

We choose a bioinformatics case (however many other domains could be, and have been, used, cf. [12,2]), in which a bioinformatician is conducting experiments to find protein sequences with interesting properties. This particular use case was chosen because it is high performance and has fine grain parallelism, which implies that recording process documentation may be difficult. Hence, showing that our approach performs in this difficult application provides support for the conclusion that the approach will work for a large set of applications with less demanding requirements.

In this section, we provide an overview of this bioinformatics example.

2.1. Biology

Proteins are the essential functional components of all known forms of life; they are linear chains of typically a few hundred building blocks taken from the same set of about 20 different amino acids. Protein sequences are assembled following a code sequence represented by a polymer (mature messenger RNA). During and following the assembly,

the protein will curl up under the electrostatic interaction of its thousands of atoms into a defined but flexible shape of typically 58 nm size. The resulting 3D-shape of the protein determines its function.

Amino acids can be *grouped* together by their chemical or physical properties. Those in the same group can often be substituted for one another in a protein sequence and the sequence will, in many cases, fold in the same manner. The ability to substitute amino acids is useful when trying to change or modify protein function. The aim of the Amino acid Compressibility Experiment (ACE) is to find other possible groups of amino acids that can be substituted for one another.

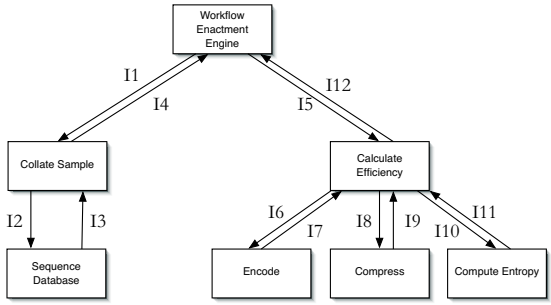
2.2. Experiment Description

In this experiment, it is assumed that protein sequences that occur in nature are efficient, i.e. they use the least number of amino acids possible to represent their function. Based on this assumption, a group is tested for interest by substituting the amino acids specified by the group with a symbol representing the group and then measuring the efficiency of the recoded sequence. The efficiency of a protein sequence can be quantified in a computational setting through compression. If a sequence compresses well, then it is not efficient, whereas if the compression causes little reduction, then the sequence is efficient. The ACE, therefore, uses compression to attempt to find possible groups of interest.

The workflow for the experiment is shown in Figure 2. It starts with the creation of a sample, which is composed from individual sequences obtained from sequence databases made available on the Web (see www.ebi.uniprot.org). This collation provides enough data for the statistical methods employed by the compression algorithms. The experiment requires that the samples be composed from dissimilar sequences. This dissimilarity is determined by using a culling service such as PISCES [17]. Once a sample is created, the symbols in it are substituted with those of a given group (Encode). This recoded sequence is then compressed with compression algorithms, e.g., gzip, bzip2 or ppmz, to obtain the length of the compressed sequence (Compress). The Shannon entropy is then computed on the recoded sequence to provide a standard for comparison (Compute Entropy). This standard removes the influence of two factors from the calculation of compressibility: the particular data encoding used to represent the groups, and the non-uniform frequency of groups. From the results, an information efficiency value is computed for the sample that is relative to both the compression method and group coding employed and takes into account the size of the sample (Calculate Efficiency). The information efficiency values for different groups can then be plotted to find those that are the largest and thus are good candidates for further investigation.

2.3. The Provenance-Aware ACE Application

Applications can be made provenance-aware by applying the PrIMe methodology [15], which analyses the application in the context of use case questions provided by the user and identifies those components in the application that should be given process documentation recording functionality. These components, or *actors* in PrIMe, record their part in the application's processes, recording p-assertions to a provenance store. Our model defines three forms of p-assertions that together provide all the information necessary to



I1: collate sample request
I2: database request
I3: sequences
I4: collated sample response
I5: calculate efficiency request
I6: sample
I7: encoded sample
I8: encoded sample
I9: compressed sample
I10: encoded sample
I11: entropy
I12: information efficiency value

Figure 2. ACE workflow.

document all the relevant information about an application's execution. The three forms of p-assertion are as follows.

Interaction p-assertions record the details of an actor's interactions — its role in the interaction (receiver or sender) the identity of the other party in the interaction and details about the data sent in the interaction.

Relationship p-assertions record the causal connections between incoming interactions and outgoing interactions, for example that a message was sent because another message was received.

Actor state p-assertions record important information the actor has access to *as it relates to an interaction*.

For example, each box in Figure 2 would be modelled as an actor and would record various different p-assertions as required in order to capture the right kinds of process documentation to answer the use case questions. In the figure, each arrow is annotated by an interaction identifier (I1 through I12), thus each would have a corresponding interaction p-assertion recorded for it as well as any necessary relationship and actor state p-assertions. Everytime the application is executed new p-assertions are created documenting that execution.

3. A Realisation of the Conceptual Model: The P-Structure

In what follows, we walk the reader though the *p-structure* — the model by which process documentation is organised in order to facilitate later querying. In Figure 3, we have the high level view of the p-structure as represented by an XML document. The fundamental method of organisation within the p-structure is the *interaction record*, thus the p-structure will contain multiple interaction records all of which together will contain the process documentation recorded for all the interactions made in the application.

```
− <ps:pstruct>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  + <ps:interactionRecord></ps:interactionRecord>
  </ps:pstruct>
```

Figure 3. The p-structure

3.1. Interaction Records

The interaction record structure contains all the information that relates to one interaction. An interaction is defined as the passing of one message from one actor to another (therefore the response to an interaction is stored within its own interaction record). Thus, the interaction record stores all the p-assertions made about that interaction from *both* parties involved in it (i.e. the sender and the receiver). [4] In our ACE example, every time an interaction occurs between two actors (for example the workflow enactment engine — termed the ACE Enactor and the calculate efficiency actor) a new interaction record would be added to the p-structure.

```
− <ps:interactionRecord>
  + <ps:interactionKey></ps:interactionKey>
  + <ps:sender></ps:sender>
  + <ps:receiver></ps:receiver>
  </ps:interactionRecord>
```

Figure 4. Interaction Records

In Figure 4, we show the XML document for an interaction record. It contains the *sender* and *receiver* views on the interaction as well as the *interaction key* — a unique identifier for this interaction. The interaction key (shown in Figure 5) contains a unique identifier for this interaction as well as information relating to the sender and receiver; specifically, in this case, we use the WS standard for addressing, which involves specify-

[4] The architecture allows for the process documentation from the sender and receiver for one interaction to be stored in separate provenance stores and thus is physically different interaction records. However, linking mechanisms enable these to be linked effectively making the physically different interaction records logically the same.

ing an *endpoint reference* that contains a URL that locates the sender/receiver. Thus, for this interaction we can see that the message source is the ACE Enactor who is sending a message to the Collate Sample actor, located by the given endpoint reference.

```
- <ps:interactionKey>
  - <ps:messageSource>
    - <ns1:EndpointReference>
      - <ns1:Address>
          http://pasoa-vmware1.ecs.soton.ac.uk/experiments/ace/AceEnactor
        </ns1:Address>
      </ns1:EndpointReference>
    </ps:messageSource>
  - <ps:messageSink>
    - <ns2:EndpointReference>
      - <ns2:Address>
          http://pasoa-vmware1.ecs.soton.ac.uk/experiments/ace/CollateSample
        </ns2:Address>
      </ns2:EndpointReference>
    </ps:messageSink>
    <ps:interactionId>e71acbcc-6e05-46d0-b2ab-a65c1d1d453d2</ps:interactionId>
  </ps:interactionKey>
```

Figure 5. An interaction key

The sender and receiver views within the interaction record each contain the sets of p-assertions made by each actor as well as their asserter information, which contains the identity of the responsible party for the p-assertions in the interaction record. In this example, this is simply their endpoint reference (see Figure 6 for the XML document representing the sender view). In this case, the sender is the ACE Enactor and so all the p-assertions within this view will originate from this actor.

```
− <ps:asserter>
  − <ns9:EndpointReference>
    − <ns9:Address>
        http://pasoa-vmware1.ecs.soton.ac.uk/experiments/ace/AceEactor
      </ns9:Address>
    </ns9:EndpointReference>
  </ps:asserter>
```

Figure 6. The Sender View

3.2. P-Assertions

Above we briefly describe the data structure that represents high level information about an interaction, such as the interaction identifier, the parties involved in the interaction as

well as their endpoint references. Apart from this information, each interaction record contains all the p-assertions made by actors for the interaction (where the p-assertions made by each actor are stored in the appropriate actor view). In what follows, we describe in detail each of the three p-assertions mentioned earlier and explain their contents.

3.2.1. Interaction P-Assertions

As previously explained, interaction p-assertions are generated for each interaction that takes place. An application will, therefore, generate sets of p-assertions from both parties involved in the interaction. As such, each interaction record will contain interaction p-assertions generated from both involved actors (i.e. one set for each *view*). Figure 7 shows the XML document for an interaction p-assertion. We can see that within this structure, there are three components. A *local p-assertion id*, a *documentation style* and a *content* element. The local p-assertion id identifies and distinguishes this p-assertion from every other that has been generated for this interaction.

```
− <ps:interactionPAssertion>
    <ps:localPAssertionId>2</ps:localPAssertionId>
    <ps:documentationStyle>http://www.pasoa.org/docstyle/AceVerbatim</ps:documentationStyle>
  + <ps:content></ps:content>
  </ps:interactionPAssertion>
```

Figure 7. An interaction p-assertion

The documentation style element, refers to one or more of a number of different ways of encoding the content of the p-assertion. Recall that interaction p-assertions include a description of the content of the message that has been sent or received. The form of this description is determined by the documentation style. The simplest documentation style is one in which the original contents of the message are simply copied verbatim within the p-assertion (this is the case with the example given in Figure 7). Other documentation styles may be used, however, to address security concerns or other non-functional requirements. For example, there are documentation styles that will encrypt the information within the p-assertion that relates to the content of the original message, or a pointer can be used to reference the original data if it is decided that the original information should not be kept with the p-assertion (for example due to space requirements). Other documentation styles are also available and descriptions of such can be found in [9].

Within an interaction p-assertion, the content element contains the payload of the message the p-assertion is documenting. As mentioned, it may have been encoded using a documentation style or it may be a verbatim copy of the information contained in the original message.

3.2.2. Actor State P-Assertions

To record actor state p-assertions, the model provides the data structure represented in Figure 8. This allows for internal data of an actor to be recorded as it refers to an interaction. The reference to an interaction is indicated by the local p-assertion id element. The *data* element is used to contain any application specific data that the actor is using

within the interaction (the example given is a *collate summary* used by the collate sample actor).

```
- <ps:actorStatePAssertion>
    <ps:localPAssertionId>2</ps:localPAssertionId>
  - <ps:content>
    - <ns7:data>
      + <ns8:collateSummary></ns8:collateSummary>
      </ns7:data>
    </ps:content>
  </ps:actorStatePAssertion>
```

Figure 8. An actor state p-assertion

3.2.3. Relationship P-Assertions

To connect outgoing interactions to incoming interactions, we provide the relationship p-assertion. This provides the means to link information arriving in an interaction with the information sent out in another information. The combination of both interaction p-assertions and relationship p-assertions gives a complete account of the data flow within an application and allows a causal graph to be generated from process documentation. Thus, working backwards from a particular result, it is possible, using both relationship and interaction p-assertions to trace back the causes of that result from earlier interactions and any associated actor state p-assertions.

```
− <ps:relationshipPAssertion>
    <ps:localPAssertionId>3</ps:localPAssertionId>
  + <ps:subjectId></ps:subjectId>
  − <ps:relation>
      http://www.pasoa.org/schemas/ace/relationships/happenedAfter
    </ps:relation>
  + <ps:objectId></ps:objectId>
  </ps:relationshipPAssertion>
```

Figure 9. A relationship p-assertion

The relationship p-assertion (see Figure 9) contains information about the id of the interaction it is associated with, the name of the relation, e.g. *caused by* or *depends upon* or, in the figure: *happened after*, as well as information relating to the inputs of the relation — called the *objects* of the relation, and the outputs — called the *subjects* of the relation.

A relation can have multiple objects but only one subject. In this way, they are very much like functions and reflect the idea that the actor is performing some operation on some input and producing an output. Figure 10 shows the data structure for the *objectID*.

This contains an *interaction key* indicating from which interaction this input was from, a *view kind*, a *local p-assertion id* identifying the relationship p-assertion to which

```
− <ps:objectId>
  + <ps:interactionKey></ps:interactionKey>
  − <ps:viewKind xsi:type="pasoa:SenderViewKind">
      <ps:description>isSender</ps:description>
    </ps:viewKind>
    <ps:localPAssertionId>1</ps:localPAssertionId>
    <ps:parameterName>request</ps:parameterName>
  − <ps:objectLink>
      <ps:provenanceStoreRef/>
    </ps:objectLink>
  </ps:objectId>
```

Figure 10. An objectId

this object id belongs, a *parameter name*, which is a name given to this *input* data reflecting the role it plays in the relationship (in the figure, the role that is played by this data item is that of a request) and, finally, an *object link*. The object link can refer to the location where the input data representing the object can be found, which may be the location of another provenance store where the actor who sent this information is sending its p-assertions.

Finally, the relationship p-assertion contains a *subjectId* element that identifies the parameter name of the data item that is the *output* of the relation (again, in this example it represents a request) as well as the id of the p-assertion it is related to (shown in Figure 11).

```
− <ps:subjectId>
    <ps:localPAssertionId>2</ps:localPAssertionId>
    <ps:parameterName>request</ps:parameterName>
  </ps:subjectId>
```

Figure 11. A subjectId

We have now completed the description of the p-structure. With the information contained in this model and with this organisation, it is a simple matter to provide components that can search the p-structure to obtain information related to provide users with information about their data. To see a detailed description of how this can be achieved see [13].

4. Performance

Enabling provenance-awareness in applications inevitably carries some overhead. In order to examine the performance implications of recording process documentation, we evaluate an implementation of the provenance store, PReServ, both independently from ACE and when integrated with it. We describe PReServ, a Java-based Web Service in detail elsewhere [11].

The experiments were run on the Iridis Computing Cluster at the University of Southampton. Iridis contains several sets of nodes (i.e. computers). The set used in the experiments consisted of 237 nodes each with two AMD Opteron processors running at 2.2 GHz and 2 GB of RAM. A local storage of 25 GB is made available as scratch disk space for running jobs. Each node has access to a shared file system where results and executables can be stored. The Provenance Service runs on a node with 4 Dual Core AMD Opteron processors running at 2.4 Ghz and 2 GB of RAM. The database stores its files on a shared filesystem with 1.4 TB of space available for the storage of process documentation. All nodes are connected by Gigabit Ethernet. Each job run on Iridis obtains an entire node for itself and due to the policy of Iridis's maintainers, a maximum of 40 jobs can be run in parallel by any one user.

One measure for accessing the performance of PReServ in a generally applicable way is to measure its throughput. In Figure 12, we see that as the number of clients increases the throughput also increases. The x-axis shows the number of jobs being run in parallel. The y-axis shows how many threads or clients to the provenance store are being run on each node and the z-axis shows the throughput in number of p-assertions per 10 minute period. Typically, a systems throughput will increase until a maxima and then throughput levels off and may slightly decrease. We attempted to find such a maxima for the provenance store by increasing the number of threads (i.e. clients) per node. However, throughput continued to increase up to the point where 32 nodes each had 16 threads recording p-assertions at the same time (i.e. 560 client connections). At this point, 234025 p-assertions (2.2 GB of p-assertions) were recorded in a 10 minute period, which means that on average a 10K p-assertion was recorded every 2.6 milliseconds.

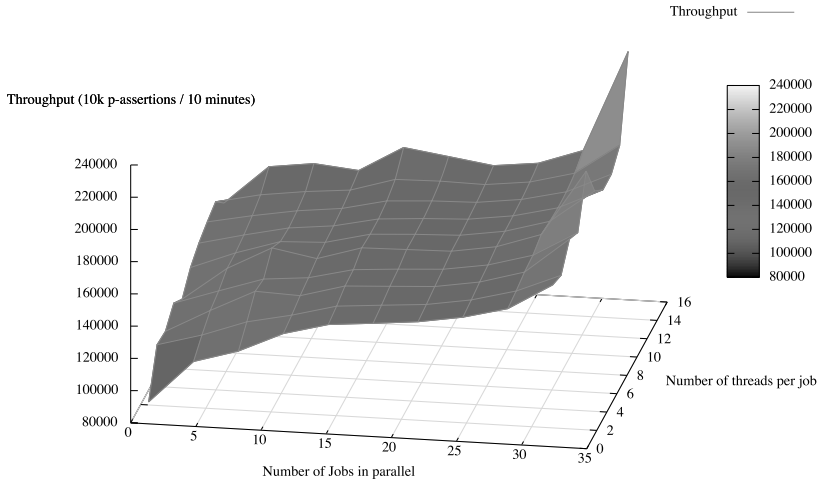


Figure 12. Throughput as the number of jobs and threads per jobs increases.

From this measure of throughput, developers can begin to ascertain the impact recording p-assertions will have on application performance. We now discuss the impact making ACE provenance-aware had on its performance.

The ACE workflow shown in Figure 2 was run as a set of 80 jobs on the Iridis cluster. Each job analysed 900 unique groups on 5 different 100K collated samples, thus,

a job generates 4500 information efficiency values. A set of 900 groups is a 50K file. Process documentation that represents the provenance of each information efficiency value is stored across two provenance stores. After one run of ACE, roughly 14 GB of information is stored in the provenance store. We measured the impact that integrating p-assertion recording with ACE had on the runtimes of ACE's component jobs. The impact on the average, maximum, and minimum job runtimes is shown in Table 1.

	Job Runtime	Job Runtime with Recording	Difference
Maximum	23:20	26:24	3:04
Average	22:24	25:17	2:53
Minimum	20:39	23:09	2:30

Table 1. Maximum, Minimum and Average job record times both with and without p-assertion recording

From this data, we conclude that there is a 13% overhead for provenance-awareness in this detailed and highly parallel use case. As such we argue that the performance drop that results in adding provenance-awareness to the application is acceptable and more than offset by the added functionality provided to the user. Moreover, because of the difficulty of ACE, this result should give confidence to developers that their applications can be made provenance-aware without unacceptable degradation in performance.

5. Related Work

The subject of provenance has not gone without notice in the literature. Under the heading of lineage, Bose and Frew present a comprehensive overview of provenance related systems [3]. Likewise, Simmham et al. give a survey of provenance in the domain of e-Science [16]. A compilation of the current state of the art is given by Moreau and Foster [14]. From an analysis of these works, we assert that the focus of provenance research has been on the implementation of concrete systems for provenance in the context of either specific domains (i.e. geographic information systems, chemistry, biology) or technologies (i.e. databases). In contrast, this work focuses on a conceptual organisation of process documentation independent of technology or domain.

Work in the database community has focused on the data lineage problem, which can be summarised as: given a data item, determine the source data used to produce that item. Cui et al. present a number of algorithms for determining the lineage of data in relational databases [6] and data warehouse environments [7]. Buneman et al. also develop a formal model of provenance for database systems that applies to both hierarchical and relational databases [4]. Our data model differs from these approaches because it can be used to represent processes that occur both inside and outside database environments.

In the e-Science community, work has focused on provenance for workflow based environments. For example, Zhao et al. present a model for provenance in the Virtual Data System (VDS), which uses the workflow graph to tie together various data elements recorded during the execution of the workflow [19]. The benefits of this approach over the p-structure is that information stating causal dependencies is inferred from the workflow and thus must not be created by the actors within the workflow. However, because a workflow is a plan and not what actually occurred, the p-structure captures the actual

causal connections according to execution, as opposed to the original workflow which implicitly captures all possible connections for all possible executions. The p-structure also differs from workflow centric systems like VDS, myGrid [18], and Kepler [1] in that it supports any type of execution environment. For example, process documentation compatible with the p-structure can be generated by Java programs, shell scripts or workflow enactment engines.

Developments in the Semantic Web community have concentrated on adding annotations to Resource Description Framework (RDF) graphs that describe the provenance of the nodes of the graph [5,8]. As with the other systems presented, these approaches are technology dependent as they rely wholly on RDF. Comparatively, the p-structure, because of its conceptual definition, can be represented using multiple technologies including RDF.

Our approach differs from all of these systems with its distinction of the *concept* of provenance from its *implementation*, as we described earlier.

6. Conclusion

In this article, we have described how the provenance open data model can be used with an example bioinformatics application. We have described how the kinds of information produced by the application can be captured in a structured way by the data model so that queries can be performed later. Having such an open data model enables it to be applied to any application domain, allowing for its wide adoption across many such domains as we described earlier. The structured format of the captured data provides many benefits that allow the model to scale up to large amounts of data that can be distributed across many provenance stores.

References

[1] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In Luc Moreau and Ian Foster, editors, *International Provenance and Annotation Workshop, IPAW 2006*, volume 4145 of *Lecture Notes in Computer Science*, pages 118–132. Springer-Verlag, 2006.

[2] S. Álvarez, J. Vázquez-Salceda, T. Kifor, L.Z. Varga, and S. Willmott. Applying provenance in distributed organ transplant management. In L.Moreau and I.Foster, editors, *LNCS: Proceedings of the International Provenance and Annotation Workshop (IPAW'06)*, volume 4145, pages 28–36, Chicago, Illinois, May 2006. Springer-Verlag.

[3] Rajendra Bose and James Frew. Lineage retrieval for scientific data processing: a survey. *ACM Computing Surveys*, 37(1):1–28, 2005.

[4] P. Buneman, S. Khanna, and W.C. Tan. Why and where: A characterization of data provenance. In *Int. Conf. on Databases Theory (ICDT)*, 2001.

[5] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM Press.

[6] Y. Cui and J. Widom. Practical lineage tracing in data warehouses. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, San Diego, California, February 2000.

[7] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. *The VLDB Journal*, 12(1):41–58, 2003.

[8] Joe Futrelle. Harvesting rdf triples. In Luc Moreau and Ian Foster, editors, *International Provenance and Annotation Workshop, IPAW 2006*, volume 4145 of *Lecture Notes in Computer Science*, pages 64–72. Springer-Verlag, 2006.

[9] Paul Groth, Sheng Jiang, , Simon Miles, Steve Munroe, Victor Tan, Sofia Tsasakou, and Luc Moreau. An Architecture for Provenance Systems. Technical report, Electronics and Computer Science, University of Southampton, 2006.

[10] Paul Groth, Simon Miles, Weijian Fang, Sylvia C. Wong, Klaus-Peter Zauner, and Luc Moreau. Recording and using provenance in a protein compressibility experiment. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC'05)*, pages 201–208, July 2005.

[11] Paul Groth, Simon Miles, and Luc Moreau. PReServ: Provenance Recording for Services. In *Proceedings of the UK OST e-Science Fourth All Hands Meeting (AHM05)*, September 2005.

[12] Guy K. Kloss and Andreas Schreiber. Provenance Implementation in a Scientific Simulation Environment. In *Proceedings of the International Provenance and Annotation Workshop (IPAW)*, pages 37–45, Chicago, Illinois, USA, May 2006.

[13] Simon Miles, Paul Groth, Steve Munroe, Sheng Jiang, Thibaut Assandri, and Luc Moreau. Extracting Causal Graphs from an Open Provenance Data Model. *Concurrency and Computation: Practice and Experience*, 2007.

[14] Luc Moreau and Ian Foster, editors. *Provenance and Annotation of Data — International Provenance and Annotation Workshop, IPAW 2006*, volume 4145 of *Lecture Notes in Computer Science*. Springer-Verlag, May 2006.

[15] S. Munroe, S. Miles, L. Moreau, and J. Vazquez-Salceda. Prime: A software engineering methodology for developing provenance-aware applications. In *Proceedings of Sixth International Workshop on Software Engineering and Middleware, SEM 06*, pages 39–46, Portland, Oregon, 2006. The ACM Digital Library.

[16] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.

[17] G. Wang and Jr. R. L. Dunbrack. Pisces: a protein sequence culling server. *Bioinformatics*, 19:1589–1591, 2003.

[18] J. Zhao, C. Goble, M. Greenwood, C. Wroe, and R. Stevens. Annotating, linking and browsing provenance logs for e-science. In *Proc. of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, October 2003.

[19] Yong Zhao, Michael Wilde, and Ian Foster. Applying the virtual data provenance model. In Luc Moreau and Ian Foster, editors, *International Provenance and Annotation Workshop, IPAW 2006*, volume 4145 of *Lecture Notes in Computer Science*, pages 148–161. Springer-Verlag, 2006.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Cyberinfrastructure and Web 2.0

Marlon E. PIERCE[a], Geoffrey FOX[a, b, c], Huapeng YUAN[a, b], and Yu DENG[a,b]
[a] *Community Grids Laboratory, Indiana University*
[b] *Department of Computer Science, Indiana University*
School of Informatics, Indiana University

Abstract. We review the emergence of a diverse collection of modern Internet-scale programming approaches, collectively known as Web 2.0, and compare these to the goals of cyberinfrastructure and e-Science. e-Science has had success following the Enterprise development model, which emphasizes sophisticated XML formats, WSDL and SOAP-based Web Services, complex server-side programming tools and models, and qualities of service such as security, reliability, and addressing. Unfortunately, these approaches have limits on deployment and sustainability, as the standards and implementations are difficult to adopt and require developers and support staff with a high degree of specialized expertise. In contrast, Web 2.0 applications have demonstrated that simple approaches such as (mostly) stateless HTTP-based services operating on URLs, simple XML network message formats, and easy to use, high level network programming interfaces can be combined to make very powerful applications. Moreover, these network applications have the very important advantage of enabling "do it yourself" Web application development, which favors general programming knowledge over expertise in specific tools. We may conservatively forecast that the Web 2.0 approach will supplement existing cyberinfrastructure to enable broader outreach. Potentially, however, this approach may transform e-Science endeavors, enabling domain scientists to participate more directly as co-developers of cyberinfrastructure rather than serving merely as customers.

Keywords. Cyberinfrastructure, Web 2.0, distributed computing, network computing

Introduction

Cyberinfrastructure 1 and e-Science 1 are conventionally presented in terms of Grid technologies 2 3 that support remote access to computational science resources (such as supercomputers), distributed data management, networked instruments and similar technologies. Web Services are a key technology for realizing this vision 4 5. In contrast to these heavyweight approaches, however, many important innovations in network programming are emerging outside the (by now) traditional Web Services framework and are collectively known as Web 2.0 6. As we discuss in this chapter, these developments need to be tracked and incorporated into the e-Science vision.

This chapter reviews some of the core Web 2.0 concepts by considering their impact on e-Science activities. For a related perspective on these issues, see 7.

Web 2.0 is not a specific set of technologies and is best characterized as a movement towards network programming for everyone. The blurred distinction between a Web application and a Web tool (such as Google's My Maps and Yahoo's Pipes applications) means that even non-programmers can make sophisticated custom Web applications. In contrast to Web 2.0, the Enterprise-centric view of Web technologies holds that Web development is the domain of highly trained programmers working in very sophisticated development environments with complicated service interface and message specifications. Enterprise technologies are characterized by various Web software vendor efforts from Microsoft, Oracle, IBM, Sun, HP, and others. Grid computing is closely aligned (through the Open Grid Forum 8, for example) with Enterprise computing. The numerous Web Service specifications (collectively known as WS-*) have formed the basis for much of Grid computing since 2001.

We compare Web 2.0 with Enterprise-style cyberinfrastructure applications in the Table 1. This table will also serve as a definition for our usage of the phrases "Enterprise Grids" and "Web 2.0" throughout this chapter.

Table 1. A comparison of Enterprise Grid and Web 2.0 approaches.

Network Programming Concept	**Enterprise-Style Grids**	**Web 2.0 Applications**
Network Services: programming interfaces and quality of service specifications.	Web Service APIs are expressed in WSDL. WS-* OASIS specifications for security, reliability, addressing, transactions, policy, etc are used. For a summary of these specifications, see 5.	Representational State Transfer (REST) services 9 are used: HTTP GET, PUT, POST and DELETE operations on URL resources are the universal API. Security is provided by SSL, HTTP Authentication and Authorization. HTTP error messages convey error conditions. No additional quality of service is provided.
Network State	WSRF 4 is used to model stateful resources.	Services are stateless (idempotent).
Network messaging	SOAP is used to convey XML message payloads. SOAP header extensions provide quality of service	Simple, more easily parseable XML formats such as RSS and Atom are exchanged. Like SOAP, Atom and RSS can include payloads of XML, HTML, etc.

Science Portals, Start Pages, and other service consumers	Science Portals are based on server-side standards such as JSR 168 portlets. Portal components are exchanged using WSRP. Science portals typically provide capabilities such as secure access to supercomputing resources.	Services are integrated using client-side applications and mash-ups 10. Self-contained JavaScript "gadgets" snippets can be embedded in Web pages. Web browser *Start Pages* aggregate RSS/Atom feeds, gadgets and widgets.
Service composition and aggregation	Workflow specifications orchestrate services and address business concerns such as support for database transactions. Scientific workflows 11 seek to capture science use cases. Workflows are typically specified with XML languages.	Mash-ups and mash-up building tools combine services into novel applications. Mash-ups are typically developed using scripting languages such as JavaScript.
Online communities	Virtual Organizations 12 are based on shared authentication and authorization systems. VOs work best for enabling cooperation between large, preexisting institutions (i.e. the NSF TeraGrid 13 or the NSF/DOE Open Sciences Grid).	Online communities are populated by volunteers and are self-policing. Work best as ephemeral, overlapping collections of individuals with shared interests.

As summarized in Table 1, there is a parallel between conventional cyberinfrastructure (Web Services, science portals, virtual organizations), and Web 2.0 (REST services, rich internet applications, online communities). Conventional cyberinfrastructure in these cases can be conceptually adapted to use Web 2.0 style approaches where appropriate. For example, not all computational services need to be associated with strong Grid-style authentication, and by loosening some of these requirements, the scientific community potentially will enable many outreach opportunities and (one would hope) more do-it-yourself Web-based computational science.

Combining Web 2.0 with conventional cyberinfrastructure also promises to enable virtual scientific communities. Grids typically are presented in terms of "Virtual Organizations", and the TeraGrid and Open Science Grid are two prominent examples. The Open Science Grid in particular is composed of a number of more-or-less dynamic virtual organizations. Grid-style virtual organizations tend, however, to focus on partnerships of real organizations that desire to share relatively scarce and valuable commodities such as computing time and access to mass storage at government-funded facilities. While this is necessary and has important consequences on security (such as the requirement for strong authentication), it is a limited (and Enterprise-centric) view of a virtual community. Web 2.0 community applications (such as Facebook, MySpaces, and other applications too numerous and ephemeral to mention 14) have demonstrated that online communities can form and gain millions of members. Flickr

and YouTube are well-known, more specialized social web sites dedicated to photo and movie sharing.

In the following sections, we review several of the technical underpinnings of Web 2.0 from the e-Science perspective. We organize these into network messaging and services; rich internet applications and user interfaces; tagging and social bookmarking; microformats for extending XHTML; a survey of Web 2.0 development tools; and a finally a comparison of Enterprise science portal development to Web 2.0-style Start Pages.

1. Network Services and Messaging in Web 2.0

1.1. Simple Message Formats

Unlike the SOAP-based Web Service specifications, many Web 2.0 applications rely upon much simpler news feed formats such as RSS 15 and (to a lesser degree) ATOM 16. These formats are suitable for simple parsing with JavaScript XML parsers. Like SOAP, both RSS and Atom carry content payloads. SOAP payloads are XML, which are drawn from another (i.e. non-SOAP) XML namespace or else encoding following conventions such as the remote procedure call convention. These encodings are specified in the Web Service Description Language (WSDL) interface, which a client can inspect in order to determine how to generate an appropriate SOAP message for the service. This can be done manually, but it is typical for a developer to use tools to hide the complications of constructing SOAP.

News feeds on the other hand convey content that can be XML but also can be text, HTML, binary files, and so forth. This is because the content is not strictly intended for machine processing, as in the case of SOAP, but also for display in readers and browsers. Interestingly, SOAP and WSDL are compatible with strongly typed programming languages (such as Java and the C/C++/C# languages), while the simpler feed formats match well with loosely typed scripting languages (JavaScript, PHP). Although there is not a strict separation (JavaScript libraries for SOAP exist, for example), it indicates the spirit of the two messaging approaches.

Atom and RSS formats are often combined with REST invocation patterns described in the next section. We give a simple Atom example in Section 1.3.

1.2. REST-Style Web Services

Representational State Transfer (REST) is a style of Web Service development that is based on manipulating URL-identified resources with simple HTTP operations (most commonly GET as well as PUT, POST, and DELETE). That is, REST services have a universal API, and effectively all the processing is done on the XML message associated with the subject URL. REST also explicitly avoids service statefulness that

occurs in more closely coupled distributed object systems. Strict REST services are idempotent: identical requests will always give identical responses.

Less dogmatically viewed, the real advantage of REST is that it provides simple programming interfaces to remote services, even though the services themselves may be quite complicated. Such issues as transaction integrity and security are not exposed in the REST interface, although they will probably be part of the interior implementation details of the service. In contrast to WSDL and SOAP, REST services do not provide service interfaces to strongly typed messages. This makes them well suited for scripting language-based clients, and they are consequently are well suited for Web developers to combine into new services and custom interfaces (called mash-ups). For a list of mash-up programming interfaces, see the "APIs" section of the Programmable Web 10. At the time of writing, more than 430 APIs are available for mashup building.

This stands in contrast with much of the Web Service based development of cyberinfrastructure. Arguably, this has been in part derived from the Enterprise-centric view that, for example, one must expose the state transitions of complicated resources on a Grid. While this may be true in some cases, it has also led to extremely complicated services and (in practice) fragile interdependencies on Web Service specifications. This also fosters the requirement for highly trained Grid specialists to implement and sustain the entire system.

The consequence of this is a division of labor in traditional Grid systems between the computer scientists and domain scientists. The computer science team members collect requirements and build systems while the domain scientists are customers to these end-to-end applications. In contrast, the REST approach combined with simple message formats potentially makes it possible for domain scientists, with general programming experience but perhaps not the specialized experience needed to build Enterprise-style services and clients, to be co-developers and not simply customers of services. That is, the domain scientists can take and use REST services to building blocks for their own custom applications and mash-up user interfaces.

1.3. An Atom Feed Example

News feeds are commonly associated with human-authored content, and the use of syndication formats distinguishes Web logs ("blogs") from simple HTML Web diaries. Feeds are associated with REST style applications since we only need standard HTTP operations (PUT and GET) to publish and retrieve feed information. In addition to human-authored content, it is also desirable to publish machine-generated data using single syndication format. This allows the feeds to be displayed in a wide range of clients (including mobile devices) with no additional development. In this section, we consider a very simple case study.

The Common Instrument Middleware Architecture (CIMA) project 17 builds Web Services and Web browser science portals for accessing both archival and real time data and metadata generated during experiments. CIMA is a general architecture, but

its primary application is to manage data and metadata generated by crystallography experiments. Besides the actual data generated by the lab, it is also desirable to monitor lab conditions such as the temperature of system components. This information is collected by CIMA services.

To implement CIMA feeds, we chose the Atom 1.0 format over RSS because Atom complies more closely with XML standards than RSS (it has a defining XML schema, for example). It also allows message payload types to be included (HTML, plain text, XML, Base64 encoded binary images, and so forth). Atom is supported by most major news reader clients. We chose the Atomsphere Java package (http://www.colorfulsoftware.com/projects/atomsphere) for implementation.

The CIMA bay temperature service is a streaming service that pushes data to a CIMA client receiver. Since news readers use HTTP GET (a "pull" mechanism), we do not encode the actual data stream in Atom as it is emitted. Instead, we create a streaming (Java) client that intercepts the stream and converts it to the Atom format. The resulting Atom feed is then returned to the news feed client using HTTP GET.

A sample Atom feed of CIMA bay temperatures is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>-5a3d6d28:1127c3baa91:-8000</id>
  <updated>2007-05-11T15:34:50.16-05:00</updated>
  <title type="text">Bay1Temp Atom Feed</title>
  <author>
    <name>Yu(Carol) Deng</name>
  </author>
  <entry>
    <id>-5a3d6d28:1127c3baa91:-7fff</id>
       <updated>2007-05-11T13:42:04.182-
   05:00</updated>
   <title type="text">SensorName, TimeStamp,
   DoubleData</title>
    <content type="html">Bay1Temp  2007-05-11
   19:34:08Z  25.5&lt;br&gt;Bay1Temp  2007-05-11
   19:34:29Z  25.5
   &lt;br&gt;Bay1Temp  2007-05-11 19:34:49Z  25.5
    </content>
  </entry>
</feed>
```

News readers vary in their tolerance of improperly formatted Atom, so it is useful to use one of the several online feed validators such as http://feedvalidator.org/, which we recommend, to check feeds for compliance. The feed listing above is shown in two different clients in Figure 1.

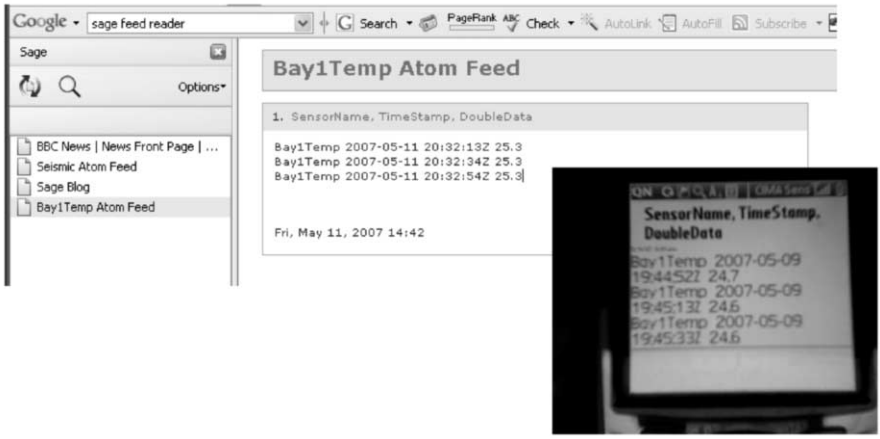


Figure 1. The Bay1 Temperature Atom feed using the Sage reader (upper left). The inset photograph (lower right) is of the same feed loaded in a cell phone's Web browser (photo courtesy of Rick McMullen).

1.4. REST for Online Computing Services: Storage and Computation

Interestingly, while computational science Grids have traditionally focused on providing access to computing, the more general online community has arguably found online storage services to be more useful. These services provide disk space for sharable online content (music and videos, for example). Amazon's Simple Storage System (S3) is a prominent example that has both SOAP/WSDL and REST style interfaces and a very simple shared secret key security model. S3 is a for-fee service, but many smaller startups provide storage services for free, if one is willing to take the risk and tolerate advertisements. Online storage systems may be used by individuals, but they also may be used by start-up social networking services as a backend storage system for shared binary content.

It is useful to consider the S3 security model through an example, which contrasts with the more complicated Public Key Infrastructure and GSSAPI based GSI model that dominates Grid systems and predates their transition to Web Services. To use S3, you must first create an account with Amazon using a credit card. As part of the registration process, you will be given a shared secret key (Amazon will have an identical copy) that can be used to digitally sign your communications with the service 26. This secret key is associated with a public identity. In communications with S3 services, the client will send the public identity of the key along with messages digitally signed by the secret key using one-way hashing (see 26 for overviews of various cryptography techniques). Amazon will use the public ID to look up its copy of the secret key and confirm the signature by reproducing the message hash. In the REST version of this service, all remote operations are performed by sending an HTTP

command: PUT the file, GET the file, or DELETE the file. Clients write the contents directly to the remote resource using standard HTTP transfer mechanisms.

Security is handled by a custom HTTP request property called "Authorization". This is a string placed in the HTTP request stream that has the form

"AWS "+"[your-access-key-id]"+":"+"[signed-canonical-string]"

The canonical string is a sum of all the "interesting" Amazon custom headers that are required for a particular communication. This is then signed by the client program using the client's secret key. Signing is performed using standard libraries, such as Java's MessageDigest class or equivalents in PHP and Ruby. Amazon also has a copy of this secret key and can verify the authenticity of the request by checking the digest value.

Your uploaded files will be associated with the URL

https://s3.amazonaws.com/[your-access-key-id]-[bucket-name]/

That is, if your key's public ID is "ABC123DEF456" and you create a bucket (a simple organizational folder) called "my-bucket", and you create a file object called "test-file-key", then your files will be in the URL

https://s3.amazonaws.com/ABC123DEF456-my-bucket/test-file-key

By default, your file will be private, so even if you know the bucket and key name, you won't be able to retrieve the file without also including a signed request. This URL will look something like this:

https://s3.amazonaws.com/ABC123DEF456-test-bucket/testkey?Signature=xxxxx&AWSAccessKeyId=ABC123DEF456

Also note that this URL can be reconstructed entirely on the client side without any communication to the serve or maintenance of a security context. All the information needed is the name of the bucket, the object key, and access to the secret key. Note even though the URL looks somewhat random, it is not, and no communication or negotiation is required between the client and Amazon S3 to create this.

Note also that this URL is in no way tied to the client that has the secret key. One could send it in email or post it to a blog and allow anyone to download the contents. It is possible to randomly guess this URL, but guessing one file URL would not help in guessing another, since the Signature field is a message digest hash of the file name and other parameters. That is, a file with a very similar name would have a very different hash value.

You can also set access controls on your files. Amazon does this with another custom HTTP header, x-amz-acl. To make the file publicly readable, you put the magic string "public-read" as the value of this header field. If your file is public (that is, can be read anonymously), then the URL

https://s3.amazonaws.com/ABC123DEF456-my-bucket/test-file-key-public

is sufficient for retrieval.

In addition to the S3 storage service, Amazon also offers an object lesson in how to build a virtual computing resource: its Elastic Compute Cloud (EC2) allows users,

for a small fee, to directly access Amazon computing resources. Such applications are well suited for pleasingly parallel applications. It is worth noting also that Amazon has deliberately made using its services as simple as possible. EC2 resources can be accessed via command line tools and ssh (in traditional Linux fashion). While we cannot comment on the viability of these as commercial endeavors, they are certainly worth using, examining, and mining for ideas. Finally, in closing, we note the Web 2.0 model at work: the Amazon S3 and EC2 service implementations are undoubtedly quite sophisticated. However, none of this sophistication is exposed in the REST programming interface.

2. Rich Internet Applications

REST services emphasize simplicity of invocation patterns and programming interface design. Sophisticated client interfaces are at the other end of the Web 2.0 spectrum. Adobe Flash plugins have shown for a number of years that the browser can overcome the limits of the HTTP Request/Response cycle to provide a more desktop-like experience for Web applications. More recently, the standardization of JavaScripts's XmlHttpRequest object (originally developed by Microsoft for Internet Explorer only but now supported by most major browsers) has enabled non-proprietary rich Web clients to proliferate. The core concept of rich user interfaces is that the Web browser can make calls back to the Web server (or Web servers) to request additional information without the user's direct request. Instead, the call-backs are driven by JavaScript events generated by the user's normal interactions with the browser user interface. This enables Web-based user interfaces to much more closely resemble desktop applications from the user's point of view.

As has been pointed out in 19, this has an important implication for Web interfaces in general and on science gateways in particular. Following the terminology of Cooper 20, traditional Web browser applications, even very sophisticated ones, are still only "transitory" applications and not "sovereign" applications such as word processor and integrated development environment (IDE) tools. Transitory applications are intended only for use for short periods of time. Web mail and Web calendar applications (and all of electronic commerce) are examples. But these are not suitable for day-long, continuous usage. In contrast, one commonly uses a Word Processor, an Integrated Development Environment tool like Eclipse, or other desktop tools for hours at a time. Rich internet applications for collaborative text editing, code development, spreadsheets, and so on are beginning to emerge and demonstrate that properly developed Web applications can indeed be sovereign.

We paraphrase the discussion of 19 here because it is an important criticism of most science gateways. Gateways have long been developed to be sovereign applications for setting up and running computational science experiments, but their success has been limited in this domain for exactly the reasons pointed out: slow response times severely limit the interactivity of the user interface, making science

portals very frustrating and limited to use as sovereign applications. Not surprisingly, most successful gateway applications have transitory interfaces: queue monitoring, job tracking, machine and network load monitoring, community data access, and so forth. The lessons and techniques of Rich Internet Applications can potentially have a dramatic impact on the next generation of gateway interfaces, allowing them to finally become sovereign applications. We now examine some of these techniques.

2.1. Ajax and JSON

Asynchronous JavaScript and XML (AJAX) is the combination of a set of pre-existing technologies that can be used to build the interactive Web client interfaces described above. AJAX's key idea is that JavaScript's XmlHttpRequest methods can be used to make calls back to the server on the user's behalf. These calls return XML messages, which can be parsed by JavaScript parsers. This latter restriction encourages the XML messages that are returned by the server to be small and uncomplicated (shallow trees suitable for stream parsing, for example). This avoids putting a computational burden on the browser and keeps the response time short to provide a higher level of interactivity. A side effect is that the XML messages, because they are simpler than is common in Web services, tend to be more human-comprehensible. RSS and Atom feeds are excellent candidates for such parsing, and streaming or pull-style parsers are much more useful than the more powerful but memory intensive and computationally demanding DOM parsers.

JavaScript Object Notation (JSON) is an alternative to XML for encoding data structures using JavaScript in over-the-wire messages. JSON objects do not contain executable JavaScript code, just JavaScript-encoded data structures. One would still need to develop a client side application with JavaScirpt and HTML to manipulate the data. XmlHttpRequest can be used to fetch JSON objects instead of XML, or JavaScript can be used to dynamically alter the HTML page to add an additional `<script>` tag. The `<script>`'s `src` attribute (normally used to download additional JavaScript libraries) can instead download JSON objects. This latter technique (or trick) provides a way to circumvent XmlHttpRequest security sandbox conventions and is known as cross-domain JSON. Instead of parsing the XML, the JSON object is directly cast to a client side JavaScript variable, after which it can be manipulated as any other JavaScript object. JSON provides a useful way to encode related data (similar to JavaBean objects and C structs) using JavaScript as the encoding format.

The primary advantage of JSON over XML is that it uses full-fledged scripting language data structures to represent data. This has the obvious advantage if one wants to pass data structures such arrays over the wire, since these are not part of the standard XML Schema. Web Services typically use SOAP XML array encoding format conventions, but these are very verbose compared to JSON arrays. The disadvantage of JSON is obviously that non-JavaScript JSON consumers and producers must parse or

translate the JSON data, although numerous such tools are available. Prominent uses of JSON include del.icio.us, Yahoo Web Services, and Google Maps.

2.2. An Example: Google Maps

The well-known Google Maps application serves as an illustration of many of the principles of rich Internet applications. The Google Maps API provides a high level, object oriented JavaScript library for building interactive maps. These libraries include convenient XML parsers, object representations of maps and overlays, access to additional services such as geo-location services, and other useful utilities. Map images are returned to the user's browser as tiles fetched from Google Map servers. Unlike older Web map applications, this map fetching is done without direct action by the user (i.e. pushing an "Update Map" button). Rather, new map tiles are returned based on the user's normal interactions with the system: panning to new areas creates events that result in new map downloads. Google Maps thus illustrates both REST style interfaces (the maps are retrieved using HTTP GET) and rich user interfaces (AJAX/JSON style JavaScript libraries that encapsulate server callbacks). The API also supports simple message formats, as developers can load and parse their own XML data for display on the maps. For a more thorough analysis of the server side of Google maps and how to build a custom tile server, see 22.

3. Tagging, Shared Bookmarking, and Folksonomies

A particularly interesting Web 2.0 development has been driven in large part by the scientific community. Connotea and CiteULike are shared bookmarking Web applications that allow users to bookmark scientific journal URLs and describe these resources with metadata. With open archives projects funded and supported by scientific funding agencies, we expect many scientific journal fields to be dramatically opened for network-based text mining and scavenging. Google Scholar and Microsoft Live Scholar already apply online search techniques for finding and ranking online publications (using Google's PageRank approach, for example, to rate the quality of publications). A large number of scholarly databases (such as the NIH's PubMed) are also available and provide service interfaces as well as web-based user interfaces.

Shared bookmarks are commonly described with simple key words called tags. Before discussing tagging in detail, we first note the related field of automatic, domain-specific mining of online literature is a very promising scientific application for Web 2.0-style applications. For example, in collaboration with the Murray-Rust group at Cambridge, we have investigated the parallelized mining of chemical abstracts using the OSCAR tool 23 to identify and convert chemical structure names in text files into SMILE identification strings. SMILE strings concisely express the two dimensional structure of the small molecule in question, and it is thereafter possible to use this to drive structure-based calculations. One may also envision useful information that may

be obtained from such mining. For instance, one may query to find all other research groups looking at chemical compounds of interest.

3.1. A Bookmarking Case Study: Del.icio.us

Del.icio.us is an example of a general purpose tagging and bookmarking service. Del.icio.us serves only as a place to store and annotate useful and interesting online web resources (URLs). Note as always that a "Web resource" can be easily extended to include not just URLs but any digital entity that can be described with a URI. That is, the digital entity does not have to be directly retrievable.

As with many Web 2.0 applications, del.icio.us comes with many different interfaces that support a wide range of interactions with the service.

- A public Web browser interface (which probably accounts for the vast majority of its usage);
- An exportable web snippet (in JavaScript) that can be embedded in other web pages such as blogs, showing personal tag and link collections ("rolls") and user information;
- Exportable RSS feeds of personal bookmarks (via http://del.icio.us/rss/your_user_name); and
- A public programming interface (described below).

Obviously, the important thing is the flexible ways for creating, accessing, and sharing information. Web 2.0 style applications are easily embeddable in arbitrary Web sites since they use the browser rather than the server as the integration point. This makes them much more flexible and simpler than Enterprise-style applications such as Java portlets and WSRP (described below). We observe that this should have a profound impact on science gateway portals.

Just as del.icio.us's simplicity for creating and delivering Web content is in stark contrast to the very heavyweight Enterprise approach, its simple approach to programming interfaces is a challenge to the complications of Web services. The service interface is briefly summarized below.

- Update: returns the update time for a user's tags.
- Tags: get and rename methods return a user's tags and rename them, respectively.
- Posts: used to get, add, or delete tag postings to del.icio.us.
- Bundles: provides programmatic access to collections ("bundles") of tags.

These service interfaces are REST-like: the API is actually a set of rules for constructing HTTP GET URLs. These URLs emit XML messages or JSON objects (whichever the developer prefers) as return values. The XML returned by these services is particularly simple and suitable for parsing by parsers available in JavaScript, PHP, and so on. Thus it is easy to use del.icio.us as an online service for storing and retrieving tags and key words specific to ones preferences. Provided one can agree with the Internet community at large on the tags (or keywords) used, it is

easy build applications that monitor del.icio.us for new resources that may be of interest and take an appropriate action. It is also easy to see how one may use such systems to build cliques, or communities, of users based solely on their shared interests.

As we have seen, the del.icio.us programming interface provides a mechanism for manipulating user or community-defined tags and tag collections. Tags are used to annotate and organize bookmarks, and they effectively define a very simple and unstructured but powerful keyword naming system. These are sometimes referred to as "folksonomies", as the keywords used to describe a particular resource are supplied by the user community. Reusing popular tags is encouraged (it makes it easy for others to find your bookmarks), effectively winnowing out redundant tags. Related tags can be collected into "bundles", but the relationship between the tags in a bundle is not explicitly defined. That is, there is no RDF-like graph, much less the logical relationships of the Semantic Web's OWL 2425.

While this is open to abuse (a resource may give itself popular but inaccurate tags to lure web traffic), the system also uses (effectively) community policing: sites with a particular keyword are ranked by the number of users who have recently added the site to their personnel collection.

4. Microformats

Microformats are an approach to making XHTML markups that separate styling and presentation from the data's meaning. Like AJAX, microformats are not a new technology but are instead the application of existing technologies in an innovative way. The key technical concept is simply to combine XHTML and tags to descriptively mark up content in ways that can be understood by humans and processed by computers. Although sometimes presented as an alternative to the Semantic Web 2425, microformats are arguably closer to providing a mechanism for implementing the famous model-view-controller design pattern within the Web browser.

While XHTML and are intended (when combined with Cascading Style Sheets) to allow custom markup definitions, the key microformats insights are

- User-defined tags can just as easily encode semantic meaning as custom formatting instructions, and
- The power of the approach comes when large communities adopt the same and conventions for common objects, such as descriptors of people, organizations, calendars, etc.

Thus the primary agenda for microformat proponents is to define small but useful markup standards (or conventions) and lobby for their widespread adoption. Prominent microformat examples include hCard and hCalendar, which are XHTML encoding variants of the vCard and iCalendar IETF specifications, respectively. The standards are simple to encode because they consist of just name-value pairs with no semantic

relationships explicitly defined. Dublin Core conventions for publication metadata would similarly be a good candidate for microformatting.

Microformats have another advantage that will probably be exploited in next-generation Web browsers: the browser can identify common or standard microformats and associate them with plugins and preferred external applications. For example, an hCalendar markup fragment may be exported to the user's preferred calendar tool.

Science portals seem to be a particularly good match for microformats for encoding scientific metadata. Obviously encoding large binary data sets using microformats is out of scope, but more realistically portals often deal with data and metadata presentation and manipulation and so would benefit from shared microformats and associated JavaScript libraries. For example, an earthquake fault can easily be encoded in as a set of name/value property values. We propose a hypothetical microformat for this below:

```
<div class="earthquake.fault">
      <div class="faultName">Northridge</div>
      <div class="latStart></div>
      <div class="lonStart></div>
      <div class="latEnd"></div>
      <div class="lonEnd"></div>
      <div class="strike"></div>
      <div class="dip"></div>
  </div>
```

This particular format, which would be embedded in an (X)HTML page retrieved by a Web browser could be associated with numerous rendering options by the page developer, hopefully chosen from pre-existing, reusable libraries and style sheets. For example, the above may be rendered (with the right helper) as either a Google Map display or a Web form for collecting user input. The user may want to alter the strike value, for example, as part of the process for setting up a finite element simulation.

5. Web 2.0 Application Development Tools

Web 2.0 applications can be built using any available tools and programming languages that have been used for Web and network programming. However, some tools have gained more popularity than others. Many of the lightweight, mash-up style Web applications are built using the so-called Linux-Apache-MySQL-PHP (LAMP) approach. MediaWiki, which powers Wikipedia, is a prominent example of a LAMP application. Ruby on Rails is another development environment that has attracted some attention for its transparent support for AJAX and built-in Object-Relational Mapping (ORM) database support. Enterprise development environments such as Java and .NET technologies are compatible with the overall Web 2.0 technologies as well. Note also that all of the above technologies are for server-side programming and generate the dynamic HTML and JavaScript needed for rich user interfaces.

Numerous tools (Ruby on Rails, Direct Web Remoting (DWR), Google Web Toolkit (GWT), and others) provide higher level libraries for generating the otherwise fragile JavaScript code needed for AJAX applications. Arguably, the primary language of Web 2.0 is really JavaScript, which has undergone a surprising resurgence, supported in large part by its use by Google and by Yahoo's YUI libraries.

Google's GWT provides an interesting example of a Web 2.0 development tool, as developers create Web applications using a Java Swing-like set of user interface classes that are (when deployed) used to generate HTML and JavaScript. The implications are interesting, as this runs counter to the conventional Model-View-Controller (or Model-2) Web application development strategies of most Enterprise frameworks. Java Server Faces (JSF) serves is perhaps the ultimate existing example of MVC web development: the user interface and code logic portions are almost completely decoupled. The assumption is that teams of developers will include both Web design specialists (who will develop the look and feel of the application using JSF XML tags) and programming specialists, who will write all of the backing Java code for interacting with databases, Web services, and other form actions.

GWT supports a very different assumption: the Web application will be designed and developed almost entirely by closely interacting groups of programmers. All HTML, links for style sheets, and JavaScript will be generated from compiled code. There is no imposed separation of responsibility among team members with different areas of specialization. Although to our knowledge no one has attempted to build a science gateway using GWT, this is arguably a more appropriate model for implementers of cyberinfrastructure.

6. From Science Portals to Widgets, Gadgets, and Start Pages

Science portals have been major constituents of cyberinfrastructure since its beginnings. Portals are extensively discussed elsewhere, and our work in the field of Grid portals is summarized in 27. Figure 2 shows a screen shot of our QuakeSim science portal to support earthquake science. QuakeSim exemplifies many of the characteristics of traditional science portals 28.

However, after examining Web 2.0 approaches, we believe that the Enterprise-based technologies used to build science portals are insufficient. In particular, Web 2.0 applications possess the following characteristics that are missing in traditional science portals:

1. Client-side integration of components from multiple service providers.
2. Multiple views of the same Web application (see for example the previous del.icio.us discussion and also Figure 3).
3. Simple programming interfaces that encourage "do it yourself" science mash-ups.
4. Widgets/gadgets that allow portal capabilities to be exported to other Web sites and start pages.

As we have mentioned earlier, many Web 2.0 web applications include embeddable JavaScript "gadgets" that enable portions of the site content to be embedded in other web pages. For example, Flickr, del.icio.us, and Connotea all have small JavaScript code snippets that users can embed in other pages.

The implications of this on science portals should be considered. Science portals tend to use the server, rather than the browser client, as the content integration point. Users may select from content provided by the server, but to add a new application or capability to the portal server requires that an entire new service be deployed. For our discussion here, we note that all the portal applications are deployed on the same server (although they are typically Web service clients to remote services). In contrast, widgets and gadgets (Figure 3) are JavaScript snippets that are embedded in the HTML, making the browser the integration point.

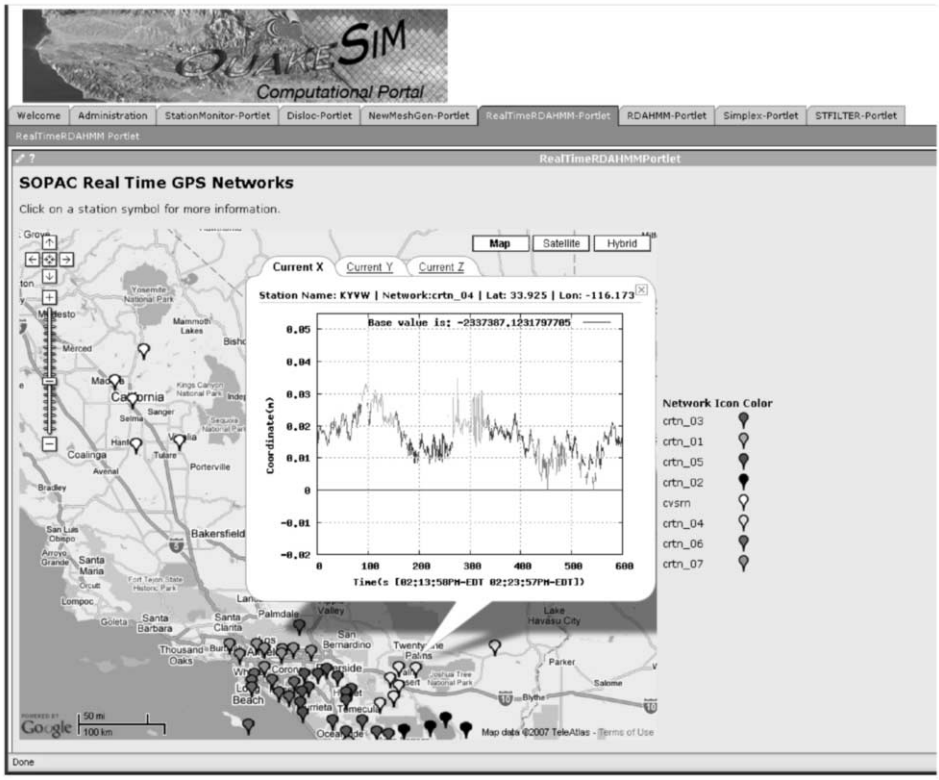


Figure 2. The QuakeSim portal screen shot shows results for the real-time analysis of Global Positioning System position data. This application is a portlet and corresponds to a web application on the portal server. Other portlets available through this portal ("StationMonitor-Portlet", "Disloc-Portlet", etc) are listed as tabs across the top beneath the logo.

Figure 3. Two del.icio.us gadget examples (inset) embedded in a blog. The gadget shows the user's latest bookmarks (top of inset) and tag cloud (bottom of inset). The blog itself generates news feeds (using Atom in this case) that can be embedded in Start Pages.

Although gadgets can be placed in any HTML page, it is common to aggregate them in personalized home pages or "Start Pages". Popular examples of start pages include Netvibes and iGoogle. Start Pages aggregate news feeds, calendars and other gadgets built by the community. We examine this process in the next section.

6.1. Creating Widgets and Gadgets: Example

We will now review the process for creating a very simple science portal gadget using Google. This can be integrated into a user's iGoogle personalized home page. We base this on our Open Grid Computing Environments portal software, which uses the portlet model illustrated in Figure 2. This gadget will manage a user's logon and open the portal as a separate window.

We first create a simple XML description for the gadget and place in a URL. For example, content of the gadget descriptor located at http://hostname:8080/gridsphere/ogcegadget.html is

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="OGCE Portal" />
```

```
    <Content type="url"
href="http://hostname:8080/gridsphere/ogce1.html" />
    </Module>
```

The content of the gadget shown here is another URL pointing to the actual HTML page that we want to load. The gadget HTML itself can include JavaScript and other markups. Once created, this gadget can be added to a Google Start Page in the usual manner (click the "Add Stuff" link).

Figure 4. A personalized Google home page with the OGCE logon gadget added to the upper left corner.

Netvibes provides a powerful alternative to iGoogle. Like iGoogle, Netvibes provides clients that can consume RSS/Atom feeds and embed externally provided widgets. Developers wishing to provide their own widgets make use of the Universal Widget API. To make a widget, one must simply provide a URL pointer to an HTML fragment that follows Netvibes widget conventions. An example listing is shown below (and see inline comments marked by <!-- -->):

```
    <?xml version="1.0" encoding="utf-8"?>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <!—Start with meta data tags-->
        <meta name="author" content="Huapeng Yuan" />
        <meta name="description" content="A Netvibes Widget
for OGCE" />
```

```
       <meta name="apiVersion" content="1.0" />
       <meta name="inline" content="true" />
       <meta name="debugMode" content="false" />
       <!—Additional meta tags can be used to control
refresh rates-->

       <!—Use Netvibes style sheets -->
       <link rel="stylesheet" type="text/css"

href="http://www.netvibes.com/themes/uwa/style.css" />

       <!—Import Netvibes JavaScript libraries -->
       <script type="text/javascript"

src="http://www.netvibes.com/js/UWA/load.js.php?env=Standa
lone">
       </script>

       <title>OGCE Portal</title>
       <link rel="icon" type="image/png"
         href="http://www.netvibes.com/favicon.ico" />
     </head>
     <!—Begin HTML web form -->
     <body>
        <form
action="http://hostname:8080/gridsphere/gridsphere?cid=log
in"
             method="post" id="form2" target="_blank">
           <p>
           <input name="JavaScript" value="" type="hidden"
/>
           </p>
           <table>
               <tr>
                   <td  style="width:100">
                       <span> UserName  </span>
                   </td>
                   <td style="width:60">
                       <input type="text" name="username"
size="20"
        maxlength="50" value="yuanh"/>
                   </td>
                   <td >
                </td>
                </tr>
           </table>
           <table>
               <tr>
                   <td style="width:100">
                       <span> Password </span>
```

```
                    </td>
                    <td style="width:60">
                        <input type="password"
name="password" size="20"
                        maxlength="50" value="qwerty"/>
                    </td>
                    <td ></td>
                </tr>
            </table>
            <table>
                <tr>
                    <td>
                        <input type="submit"
name="gs_action=gs_login"
                         value="Login"/>
                    </td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

The example listing is standard (X)HTML for a Web form, with some additional Netvibes meta tags that provide necessary metadata about the widget. The basic steps are

1. Develop a widget as a standalone XHTML page.
2. Add Netvibes-required meta tags to the HTML as shown.
3. Use Netvibes-provided CSS and JavaScript libraries.
4. Write (if desired) control code for the widget's behavior using JavaScript libraries provided by Netvibes.

The final step is not used in our example (which invokes a login form) but in general one would use Netvibes JavaScript libraries to control newsfeed and (more generally, AJAX or JSON) data loads, parse XML responses, cast JSON code, and other JavaScript tasks as discussed previously. Note our simple HTML form-based example will actually redirect you to the page specified by the action attribute, so to keep actions within the user's start page, you must process the user's actions with JavaScript.

7. Conclusions

We have reviewed Web 2.0 technologies with an attempt to show how they can be used and combined with Cyberinfrastructure and e-Science. As the term e-Science implies, much work on cyberinfrastructure has focused on Enterprise-style applications

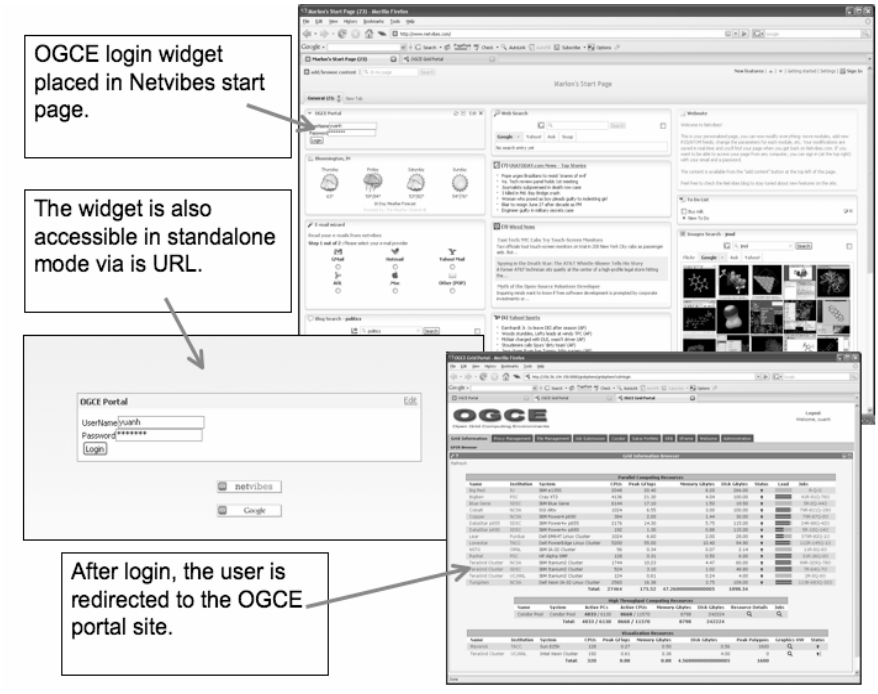


Figure 5. The Netvibes login widget API can be used to create custom portal widgets.

that we surveyed in column 2 of Table 1. The Enterprise approach may be roughly characterized as requiring specialized programming and development expertise with particular tools (such as the Globus toolkit or the OGSA-DAI database services) and places an emphasis on sophistication (such as complicated XML specifications, security concerns, reliability, database transaction support, etc.).

With some reservations on coining yet another term, we propose i-Science as an alternative to e-Science that uses lessons learned from Web 2.0. i-Science is network based and should encourages "do it yourself" Web science applications. i-Science project teams should closely integrate domain scientists as co-developers and not simply treat them as customers and requirement sources for the IT development team. To accomplish this, services and components must hide complexity, rather than expose it. Interfaces and message formats must be simple, so that anyone with a reasonable amount of programming skill can learn to develop applications that add value.

8. Acknowledgements

We thank Rick McMullen and Kia Huffman of the CIMA project for help creating the Atom feeds. This work was partially supported by the national Science Foundation, Award Number SCI-0537498.

References

1. D. E. Atkins, K. K. Droegemeier, S. I. Feldman, H. Garcia-Molina, M. L. Klein, D. G. Messerschmitt, P. Messina, J. P. Ostriker, and M. H. Wright, "Revolutionizing Science and Engineering Through Cyberinfrastructure." Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure, January 2003. Available from http://www.nsf.gov/cise/sci/reports/atkins.pdf.
2. A. J. G. Hey, G. Fox: Special Issue: Grids and Web Services for e-Science. *Concurrency - Practice and Experience* 17(2-4): 317-322 (2005).
3. F. Berman, G. Fox, and T. Hey, T., (eds.). *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, Chichester, England, ISBN 0-470-85319-0 (2003). http://www.Grid2002.org.
4. I. Foster and C. Kesselman (eds.) *The Grid 2: Blueprint for a new Computing Infrastructure*, Morgan Kaufmann (2004).
5. I. Foster., "Globus Toolkit Version 4: Software for Service-Oriented Systems." *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2-13, 2006.
6. M. P. Atkinson, D. De Roure, A. N. Dunlop, G. Fox, P. Henderson, A. J. G. Hey, N. W. Paton, S. Newhouse, S. Parastatidis, A. E. Trefethen, P. Watson, and J. Webber: Web Service Grids: an evolutionary approach. *Concurrency - Practice and Experience* 17(2-4): 377-389 (2005).
7. P. Graham (November 2005). Web 2.0. Available from http://www.paulgraham.com/web20.html.
8. D. De Roure and C. Goble, "myExperiment – A Web 2.0 Virtual Research Environment". To be published in proceedings of *International Workshop on VirtualResearch Environments and Collaborative Work Environments*. Available from http://www.semanticgrid.org/myexperiment/myExptVRE31.pdf.
9. The Open Grid Forum: http://www.ogf.org/
10. R. T. Fielding, and R. N. Taylor: Principled design of the modern Web architecture. *ACM Trans. Internet Techn.* 2(2): 115-150 (2002).
11. For a list of public mash-ups, see The Programmable Web: www.programmableweb.com.
12. G, C. Fox, D. Gannon: Special Issue: Workflow in Grid Systems. *Concurrency and Computation: Practice and Experience* 18(10): 1009-1019 (2006).
13. I. T. Foster, C. Kesselman, and S. Tuecke: The Anatomy of the Grid - Enabling Scalable Virtual Organizations *CoRR* cs.AR/0103025: (2001).
14. C. E. Catlett: TeraGrid: A Foundation for US Cyberinfrastructure. *NPC 2005*: 1. See also http://www.teragrid.org/.
15. For a list of social networking Web sites and their claimed memberships, see http://en.wikipedia.org/wiki/List_of_social_networking_websites.
16. D. Winer, "RSS 2.0 Specification". Available from http://cyber.law.harvard.edu/rss/rss.html.
17. M. Nottingham and R. Sayre (Eds), "The Atom Syndication Format." *Internet Engineering Task Force RFC 4287.* Available from http://tools.ietf.org/html/rfc4287.
18. R. Bramley, K. Chiu, T. Devadithya, N. Gupta, C. A. Hart, J. C. Huffman, K. Huffman, Y. Ma, and D. F. McMullen: Instrument Monitoring, Data Sharing, and Archiving Using Common Instrument Middleware Architecture (CIMA). *Journal of Chemical Information and Modeling* 46(3): 1017-1025 (2006).
19. J. J. Garrett, "Ajax: A New Approach to Web Applications." Available from http://www.adaptivepath.com/publications/essays/archives/000385.php.
20. D. Crane, E. Pascarello, and D. James, *AJAX in Action*. Manning, Greenwich, 2006.
21. A. Cooper, "Your Program's Posture." Available from http://www.chi-sa.org.za/articles/posture.htm.

22. D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)". *Internet Engineering Task Force RFC 4627*. Available from http://tools.ietf.org/html/rfc4627.
23. Z. Liu, M. E. Pierce, and G. C. Fox Implementing a Caching and Tiling Map Server: a Web 2.0 Case Study Proceedings of the *2007 International Symposium on Collaborative Technologies and Systems* (CTS 2007).
24. P. Corbett, and P. Murray-Rust: High-Throughput Identification of Chemistry in Life Science Texts. *CompLife* 2006: 107-118.
25. The World Wide Web Consortium Semantic Web Activity: http://www.w3.org/2001/sw/.
26. N. Shadbolt, T. Berners-Lee, and W. Hall: The Semantic Web Revisited. *IEEE Intelligent Systems* 21(3): 96-101 (2006).
27. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, New York, 1996.
28. J. Alameda, M. Christie, G. Fox, J. Futrelle, D. Gannon, M. Hategan, G. von Laszewski, M. A. Nacar, M. Pierce, E. Roberts, C. Severance, and M Thomas. The Open Grid Computing Environments collaboration: portlets and services for science gateways. *Concurrency and Computation: Practice and Experience* Volume 19, Issue 6, Date: 25 April 2007, Pages: 921-942
29. N. Wilkins-Diehr and T. Soddemann: Science gateway - Science gateway, portal and other community interfaces to high end resources. *Proceedings of Supercomputing 2006*: 16.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

BabelPeers: P2P based Semantic Grid Resource Discovery

Dominic BATTRÉ [a], Felix HEINE [a], André HÖING [a] and Odej KAO [a]
[a] *TU Berlin, Complex and Distributed IT Systems*

Abstract. In this chapter, we describe the BabelPeers project. The idea of this project is to develop a system for Grid resource description and matching, which is semantically rich while maintaining scalability and reliability. This is achieved by the distribution of resource data over a p2p network, combined with sophisticated mechanisms for query processing, reasoning, and load balancing.

We start by describing the benefits of semantically expressive languages for Grid resource description, and then continue to explain our methods, and how they help to maintain scalability. This includes distributed data storage and query processing strategies. Reliability is given by replication in the p2p network. Special emphasis is given on novel methods for load balancing and efficient query processing. Finally, we present benchmarks and simulations to show the good performance of the BabelPeers system.

Keywords. Grids, Semantic Grid, Resource Discovery, Peer-2-Peer, RDF, SPARQL

Introduction

Resource discovery in large Grid systems is a challenging task. First, the issue of **scalability** gets increasingly urgent, influenced by various factors. The larger the Grid grows, the more resource descriptions must be stored, and the more users are querying the system at the same time. Furthermore, as modern Grids go beyond pure sharing of compute power and storage capacities, many different resource types are present. These include sensors, complex scientific instruments, databases, and arbitrary services. Thus, also the complexity and level of detail of the resource descriptions increases. And, last but not least, users are interested not only in individual resources, but also in sets of resources which interoperate and are able e. g. to execute a workflow. This means that a good resource discovery system must have a complete view of all resources of the Grid in order to find a combination of matching resource residing at various providers.

Second, as also indicated in the previous paragraph, the formalism to describe resources must be **highly flexible**. With an increasing variety in resource types, resource description standards like the Glue Schema [3] are no more sufficient. More problematic, no single standard will be able to keep pace with the new developments, like e. g. FPGA boards or other special resource types. Thus flexibility in the resource description framework is needed that allows one to add new types of resources on the fly. To ensure interoperability despite these dynamics, it is necessary to have a meta-language that allows encoding relationships between new types that have been added later. Ontology frameworks from the Semantic Web [7] are ideal for this purpose.

The **BabelPeers project** aims to provide a system that resolves these two main requirements. For scalability, it uses a peer-2-peer (p2p) network based on distributed hash tables (DHT) to store and query the resource information. For flexible resource descriptions, it uses the resource description framework (RDF) from W3C. BabelPeers also supports RDF Schema inheritance to integrate multiple possibly incompatible resource description schemas.

Main parts of the system are a data dissemination algorithm that places the resource information on specific nodes and ensures fault-tolerance through replication, an efficient query algorithm that allows querying the aggregated information of all participating peers, an RDF Schema processor that evaluates the RDF Schema rules, and a load-balancing mechanism that ensures good performance of the system. Additionally, we have started to develop a wrapper for BabelPeers that allows us to use BabelPeers as a replacement for the Globus WS-MDS component. We have evaluated the performance of BabelPeers both through simulations and real benchmarks with up to 128 nodes.

This chapter is organized as follows: After describing related work, we explain how RDF and RDF/S can be used to describe resources and how the matching process can benefit from the features of this language. In section 3 we describe the technical details how BabelPeers works. This includes the data dissemination, the query processing and the load balancing features. Furthermore, this section describes the WS-MDS interface for BabelPeers. In section 4 we present our benchmark and simulation results, and finally we conclude in section 5.

1. Related Work

We have to compare our system both with existing matchmakers from the Grid world, as well as with RDF systems and P2P research. We start with Grid matchmakers.

One of the earliest matchmakers is the **Condor ClassAd-System** [22]. It employs the so-called ClassAd language, which uses mainly pairs of attributes and values, enhanced with a rich language for expressions over the attributes. The system has been extended to support simultaneous matchmaking of multiple resources as a single atomic operation [21].

Within the Globus Toolkit, matchmaking is done by the **WS-MDS** component (Monitoring and Discovery Services) [12]. WS-MDS resource indices can be organized hierarchically. Thus services which are closer to the root of the hierarchy contain information about a broader set of resources, however this information will be more out-dated and probably not as detailed as information sources at the leaves. Another Grid resource discovery system which is based on distributed hash tables is proposed in [23].

Several RDF triple stores emerged in the recent past. Very prominent are Sesame [9] and Jena [24] as centralized triple stores. In order to improve scalability several other projects use distributed hash table P2P networks (DHTs). Among these are RDFPeers [11], Atlas [17], RDFCube [19] and GridVine [1]. While the projects share common objectives they differ in their query processing and load-balancing strategies and capabilities. Edutella [20] follows a different route because it uses super-peer P2P networks as an underlying architecture.

Load balancing plays an important role for RDF triple stores that use Chord-like DHTs. As explained in [5], popular approaches like power of two choices [10] and simple

moving of nodes [25] in the DHT do not work well in this case. Ganesan et al. focus on load balancing and range queries in p2p systems in [13]. Their approach is based on SkipNets [14]. It is comparable to our load balancing approach, however, they do not regard query load balancing. On the other hand, they employ a mechanism to move nodes in the network to new positions, which helps to reduce the load balancing overhead. We aim to combine this method with our approach in the next BabelPeers version.

A novel approach to reasoning in p2p systems is presented in [2] where the information is distributed in a much coarser granularity.

2. Semantic Resource Discovery

In this section, we describe how we model resource description and discovery with RDF. Although we focus on traditional Grid resources like high performance cluster computers or large scale storage resources, our system is open to match anything which can be described using RDF.

RDF can also be used to encode existing standards like the Glue information model [3]. Glue is based on classes and attributes for these classes. Thus it can be translated straight-forward to RDF and RDF Schema. After translating the schema, individual resources can be described in RDF. Objects are RDF resources, their attribute values and the relations to other objects are encoded using RDF triples with predicates from the schema. However, after this translation we can start to use the power of RDF to achieve an *enhanced resource discovery process*. We can use generic knowledge to enhance the matchmaking process.

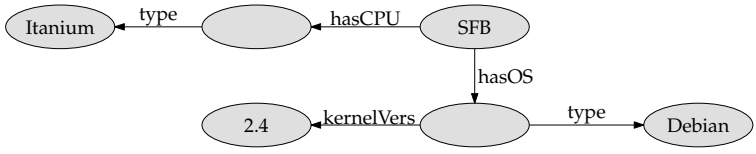
2.1. Example

RDF descriptions consist of a set of *triples*, whose elements are called *subject*, *predicate*, and *object*. The triples can be read like sentences. A set of triples can also be regarded as a graph, where each predicate is a directed edge connecting the subject vertex with the object vertex.

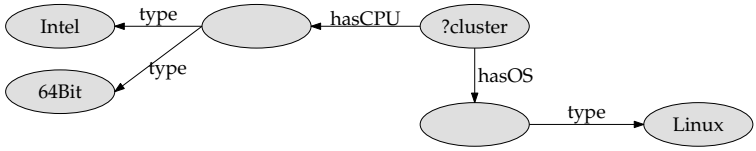
We start with an example how we use RDF [18,6] in combination with RDFS [8] to describe resources and background knowledge about resources. For simplicity, we use a reduced information model, and we omit namespaces. The techniques shown here apply to other models as well.

Consider a cluster named SFB running the Debian distribution with kernel version 2.4 of the Linux operating system. The cluster nodes are equipped with Itanium processors. A part of the RDF graph describing this system is shown in figure 1(a). Both, the operating system and the processor are described by blank nodes as we are not interested in these entities themselves but rather in their type and their properties.

Queries can be modeled as graphs, too. A simple query graph is shown in figure 1(b). It looks similar to the first graph, but there are some differences. First, the entity which is queried is denoted with a question mark. Second, the type of the operating system is specified as `Linux` compared to `Debian` in the resource description. Third, the type of the processor is specified as a combination of two types: `Intel` and `64Bit`. Thus, the query is more generic than the description. The entire query graph should be read as a template which is to be matched with a subgraph of the resource description graph.



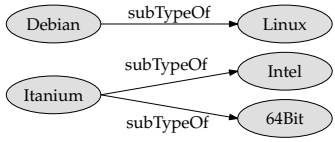
(a) Resource Description Graph.



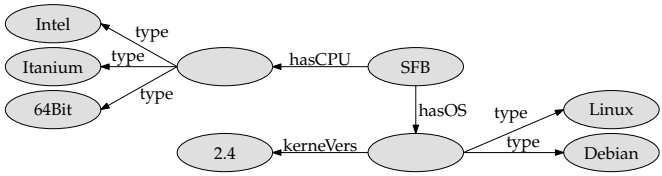
(b) Query Graph.

Figure 1. Resource Description with RDF.

In order to be able to match this query with the resource description, we need additional background knowledge, which we encode in RDF Schema. It encompasses information like `Debian` is a subclass of `Linux`. Figure 2(a) shows how this knowledge is encoded in RDF/S.



(a) Graph for Schema Knowledge.



(b) Model Graph after Applying RDFS Rules.

Figure 2. Resource Matching with RDF Schema.

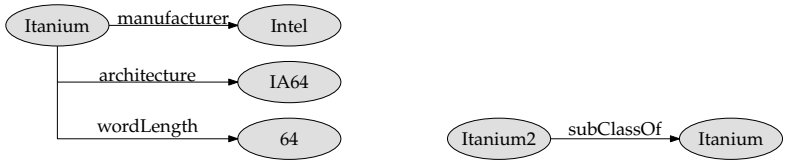
The background knowledge is integrated in the query process by applying the RDF Schema entailment rules [15]. These rules are executed in a forward-chaining manner, thereby generating new triples. For instance, entailment rule "rdfs9" of [15] generates new `rdf:type` edges. In the resulting graph, shown in figure 2(b), a subgraph can now be found which matches the query graph. The answer to the query is the `SFB` entity.

2.2. Integrating Background Knowledge

The introductory example already shows one type of background knowledge which is very important. Information about the class hierarchy can be used during the matchmaking process. This allows locally extending the class hierarchy, including very specific

classes for entity types that are elsewhere unknown. Through the RDF Schema mechanism, these entities are also published to be objects of more generic types, thus being discovered by queries searching for the generic entities.

RDF does not distinguish between classes and instances and therefore allows describing both identically with triples. Thus, generic information about types can be encoded as well. In figure 3(a), the *class* `Itanium` is used as subject and several triples describe further information that hold for every Itanium processor. This approach is an alternative to the way the same information has been encoded in the above example, allowing a clearer distinction between several aspects.



(a) About the Itanium Processor. (b) About the Itanium2 Processor.

Figure 3. Background Information.

This way of modeling provides for example the possibility to use filter expressions in queries that restrict suitable word lengths of processors. If the word length is encoded as a class called `64Bit` as shown in figure 2(a), queries for CPUs with a word length greater or equal that 32 bit are not possible without listing every word length explicitly or modeling this by `subClassOf` relations. In figure 3(a), the word length is an integer literal, which can be used to filter the results.

These two options can also be combined together. Consider e. g. a processor class hierarchy where Itanium2 is defined to be a subclass of Itanium, see figure 3(b). A provider describes its resource to have an Itanium2 processor, see figure 4. Now look at the query in figure 5. It asks for a cluster with a processor that is manufactured by Intel.

Figure 4. Resource with Itanium2 Processor.

How can this query be answered? First, the RDF Schema rules use the `subClassOf` relationship defined in figure 3(b) to generate a new triple stating that the CPU of `ClusterX` is also of type `Itanium`. This new triple provides a link between `ClusterX` and the background knowledge for Itanium shown in figure 3(a), which includes the manufacturer `Intel`. Summarizing, the query is answered using three steps: First, three different RDF fragments are combined. Second, RDF Schema reasoning is applied to the union of these fragments. Third, the query pattern is matched to the resulting RDF graph.

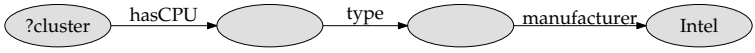


Figure 5. Query for a Cluster with an Intel Processor.

This example shows an important feature needed for resource discovery. The system must be able to combine information originating from various sources, as the pieces of information which have been combined might reside on different nodes in the network. This is a main motivation for the system design of BabelPeers as described in the next section.

3. System Overview

In this section, we describe the methods and algorithms used in the BabelPeers system. This includes the way we disseminate the information, the query processing algorithms, and load balancing issues.

3.1. Data Dissemination

As explained in section 2, we aim to integrate knowledge (in the shape of triples) originating from different sources, i. e. different nodes in the p2p network. Even broadcasting a query to all nodes in the network and collecting the results could not deliver the same quality of results one gets if the knowledge is integrated. Thus we need to disseminate the knowledge to well-defined nodes in order to be able to find and access it efficiently during query processing.

For this purpose, we use a p2p network based on distributed hash tables (DHT). In a DHT, each data item is associated with an identifier from an identifier space, e. g. $0, \ldots, 2^{128} - 1$. Each node in the network is responsible for a certain range of this identifier space. Every item to be stored is then pre-distributed to the node responsible for the identifier of the item. For fault tolerance, items are additionally replicated over multiple nodes. In our case where data items are RDF triples, we disseminate each triple to three different nodes based on its components subject, predicate, and object. Thus we can later access the triples even if only one of the components is known.

For the RDF Schema reasoning, we follow a forward chaining approach. This means that we generate and store instances of every new triple which follows from the RDF Schema rules, like the additional `type` triples in the previous section. Our dissemination scheme has the advantage, that all triples which are needed to do this forward chaining will be located on the same node. Thus, after dissemination, we can run the reasoning process on each node locally, generating the new triples. However, these newly generated triples are then disseminated to the network to be accessible via the standard indices over subject, predicate, and object.

The whole process is visualized in figure 6. Each node has some triples in its "local triples" store. These triples are disseminated via the p2p network to the responsible nodes, which store them in their "received triples" store (step 1). Using only the triples in the received store, RDF Schema reasoning is performed to generate new triples locally (step 2). The new triples are stored in the "generated triples" store, and then disseminated again over the network (step 3). At the target nodes, they are again stored in the received store, where they might fire new RDF Schema rules. The whole process terminates as soon as no more rules can be fired.

Updates and deletions are handled via a soft-state process. Thus, every triple carries an expiration time and is automatically removed from the network when it does not

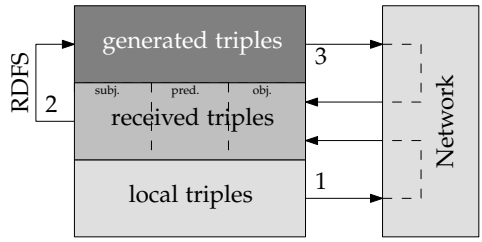


Figure 6. Triplestores.

get refreshed. This means that the dissemination process runs periodically to keep the knowledge in the network up-to-date.

3.2. Query Processing

We provide two different kinds of query processing strategies. The goal in the first case is an exhaustive query evaluation, i. e. an evaluation that returns all matches to a query in the RDF graph. Depending on the type of query, you might get enormously large answers from a resource discovery system. If you ask a very generic query, e. g. by asking for resources with a Unix-type operating system, it is likely that you receive a really large answer list. Just like using Google to search for information in the WWW, you will not look at every answer in detail, but only at a few of them. To support this type of query, we developed a second strategy named top k that retrieves efficiently only k answers instead of generating an exhaustive list of all answers. The answers can be ranked according to an attribute of the resources. This is useful when you are e. g. interested in all Linux servers in the Grid, but look for the server that has the largest main memory.

3.2.1. Exhaustive Query Evaluation

In the previous section we have given an intuitive definition of RDF graphs. The graph that contains the resource description is called a model graph. Each labeled, directed edge can be represented as a triple $t = (s, p, o)$. Its components are called subject, predicate, and object respectively. A subject can be an RDF URI reference or a blank node; a predicate has to be an RDF URI reference; an object can either be an RDF URI reference, a blank node, or a literal. For simplicity we assume that each triple t is element of $\mathcal{X} \times \mathcal{X} \times \mathcal{X}$. The set of all triples in the network is denoted with T_M.

A query as described in figure 1(b) can be modeled as a set of answer-variables and triple patterns. Following [17], we define a *conjunctive query* Q as a formula

$$\underbrace{?x_1, \ldots, ?x_n}_{H(Q)} \text{ :- } \underbrace{(s_1, p_1, o_1) \wedge (s_2, p_2, o_2) \wedge \cdots \wedge (s_m, p_m, o_m)}_{T(Q)}$$

where $?x_1, \ldots, ?x_n$ are variables and each (s_i, p_i, o_i) is a triple pattern, consisting of URIs, blank nodes, literals, and variables. $H(Q)$ denotes the head of the query, the set of variables that will appear in the answer set. Each variable $?x_k$ appears in at least one triple pattern. The triple patterns (s_i, p_i, o_i) of the query make up the set $T(Q) \subseteq (\mathcal{X} \cup \mathcal{V}) \times (\mathcal{X} \cup \mathcal{V}) \times (\mathcal{X} \cup \mathcal{V})$, where $\mathcal{V}$ denotes the set of variables. $T(Q)$ is also called the query graph. Note that it can contain variables that do not appear in the query head.

The goal of the query processing is to find all assignments of variables to URIs, blank nodes, and literals such that all triples of the query can be found in the model graph, or formally: Find all valuations $v : \mathcal{V} \mapsto \mathcal{X}$ such that $T(Q)[?x_1/v(?x_1), \ldots, ?x_n/v(?x_n)] \subseteq T_M$.

Example: Consider the model graph of figure 1(a) and assume that background knowledge (given in RDF Schema) augmented this graph by information such as "Itanium processors are 64 Bit processors by Intel" and "Debian is a Linux distribution." The query depicted in figure 1(b) could be modeled as

$$?cluster \text{ :- } (?cluster, \text{hasCPU}, ?v_1) \wedge (?v_1, \text{type}, \text{Intel}) \wedge (?v_1, \text{type}, \text{64Bit}) \wedge (?cluster, \text{hasOS}, ?v_2) \wedge (?v_2, \text{type}, \text{Linux})$$

A valid valuation given the extended model graph would be $v = \{(?cluster, \text{SFB}), (?v_1, \_:1), (?v_2, \_:2)\}$ where _:1 and _:2 represent the blank nodes in the model graph.

Once we have found valid valuations, we can project them to only those variables which appear in the head of the query. These variables are denoted with $H(Q) \subseteq \mathcal{V}$.

Focus of this section is how to find valid valuations. The query processing described in this section is similar to work presented by Liarou et al. in [17] but extended by sophisticated means to determine the order in which triples of the query are processed (see [16]). These means are crucial to the overall performance of the system. The general idea of the query processing strategy is to calculate the result of a query iteratively. We start with a single triple pattern of the query graph $T(Q)$ and do a lookup for possible valuations of the variables occurring in this triple pattern. Then we extend this intermediate result by doing a lookup for a second triple pattern and joining the results. We execute this operation until all triple patterns have been regarded.

Assume that the triple patterns in $T(Q)$ can be ordered by a heuristic such that triples patterns which have few matches are processed first. This heuristic will be described later on. We denote with $next(T(Q))$ the triple pattern of $T(Q)$ that shall be processed next.

Triple patterns can be processed if they contain at least one fixed value (URI, blank node, or literal, but not a variable) or a variable for which we have determined possible valuations already. We denote with $\sigma_{next(T(Q))}(T_M)$ the selection of triples from the model graph that match the next triple pattern to be processed.

During query processing, intermediate results are stored in a relation R, whose columns store the valuations of variables. Initially, R is empty.

Simplified, with this, we can process a query by applying the following step iteratively:

$$R' = \pi_V \left(R \bowtie \pi_{\text{var}(next(T(Q)))} \left(\sigma_{next(T(Q))} (T_M) \right) \right)$$

where V is the set of variables occurring in the head $H(Q)$ of the conjunctive triple pattern q or in triple patterns of $T(Q)$ that have not been processed yet.
The operations are (beginning from the inner most operation):

1. query those triples from the model graph that match the triple pattern $next(T(Q))$
2. project the result to only the variables of the triple pattern $next(T(Q))$
3. calculate the natural join with the previous results
4. project the result to only those variables (columns) that are relevant for the result or the further processing of the query

Once all triple patterns have been processed, relation R' contains the result of the query.

The query evaluation can be accelerated significantly by reducing the network traffic caused by triple lookups. As explained, each triple pattern of the query consists of RDF URI references, literals, and variables. A lookup of a triple pattern consisting of three URI references can return at most one triple. A lookup of a triple pattern consisting of two URI references and a variable can, however, return a large number of triples. If we restrict the triple patterns to be returned to only those that may be relevant in the future, this can save a lot of bandwidth. When a variable occurs the first time in a lookup we receive a set of possible valuations for this variable. As triple patterns are conjunctive, this set of possible valuations can only be pruned but never be extended by future lookups. Therefore, we encode the set of possible valuations with Bloom filters while doing lookups and request only those triples that fall into our existing candidate set.

Furthermore, we improve the query processing throughput by first determining for each triple pattern that we can process in the next step, how many matching triples are expected. Then we continue with the triple pattern that delivers the least results. Using this order creates small candidate sets for variables and helps retrieving smaller triple sets with future lookups. These strategies are described in detail in [16].

3.2.2. Top k Query Processing

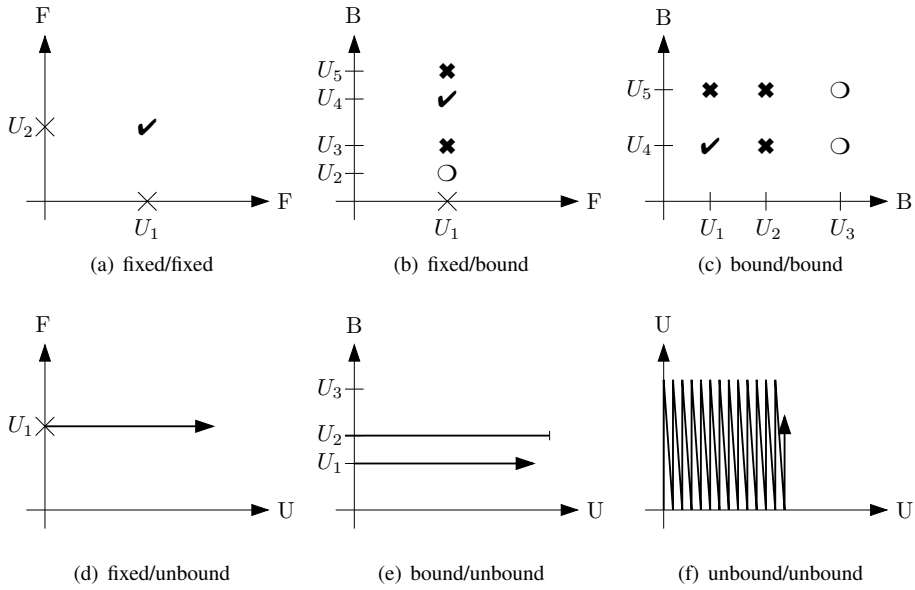
As motivated before, the top k query processing strategy serves the purpose of retrieving only the best k matches (e. g. the fastest clusters that run Linux). Query processing runtime is reduced because not all matches need to be generated.

The general idea of the evaluation function is to iterate over all possible assignments to triples, assume one, and proceed to a recursive evaluation until we encounter contradictions, find a complete match, realize that we have found a sufficient number of matches, or until we cannot assign any more triples. So far, this is straight forward. The crucial part is to find the next possible assignment (called *candidates*) for a given query triple. If this is done on a triple by triple basis, the algorithm results in sending a huge number of very small messages over the network, each one collecting a single triple.

In order to avoid this situation, we developed a combined **caching and look-ahead** mechanism. The look-ahead tries to guess which candidates might be used next in the backtracking and to fetch a larger set of these candidates. At the same time, old candidates might be reused during the backtracking and will thus be cached to avoid refetching over the network.

Each query triple consists of three components, which might either be fixed values or variables. Each variable in a query triple might either be *bound* or *unbound*. A variable is bound if it was seen before in a higher recursion level, and unbound if it occurs for the first time. We split the triple into one component which defines the key of the DHT and two remaining components. If possible, we choose a fixed URI as DHT key; otherwise we have to use a variable which is already bound to some value. For the other two components of the query triple, we can encounter six different cases:

1. two unbound variables,
2. an unbound variable plus a bound variable,
3. an unbound variable plus an fixed URI or literal,



(a) fixed/fixed (b) fixed/bound (c) bound/bound

(d) fixed/unbound (e) bound/unbound (f) unbound/unbound

Figure 7. Cache Types

4. two bound variables,
5. a bound variable plus an fixed URI or literal, and
6. two URIs / literals

For each of these cases we define a specially optimized type of cache. These caches are depicted in figure 7.

The caches have to query the next chunk of up to c candidates for a query. They deliver these chunks triple by triple. For scalability reasons, the peer which will process the query does not store any state information, so the requesting peer is in charge of submitting the state along with the actual request. The state can consist of the set of triples we want to gather information about (first three cases below) or of a set of markers, which define the last triples for which we know information already (last three cases below).

The simplest cache for **fixed/fixed** components (see figure 7(a)), which occur if a RDF query contains a triple with three URIs, does a simple lookup without look-ahead. The state of the cache can be "triple exists in RDF graph" (represented by a check mark in the figure), "triple does not exist in RDF graph" (cross), or "unknown whether triple exists in RDF graph" (circle). For **fixed/bound** component pairs (see figure 7(b)) the caching is simple as well. A peer requests a chunk of triples by specifying the fixed component and a set of candidates for the bound component for which it wants to retrieve the state. For **bound/bound** components (see figure 7(c)) we build up a request containing a set of unknown combinations of already known values for the bound variables. The **fixed/unbound** cache (see figure 7(d)) is similar to the fixed/bound cache, except that it is sufficient to request the next c elements starting after a given position. Therefore, we submit the fixed element and the last inspected value for the unbound element as the request. The **bound/unbound** cache (see figure 7(e)) extends this by storing and sub-

mitting markers for the last known elements in several rows. The peer who processes a request starts sending triples at the first marker until c triples have been sent or continues at the next marker if the row (candidates) do not provide c triples. For the **unbound/unbound** cache (see figure 7(f)) it is again sufficient to submit a single marker which determines the next triples to be delivered.

3.3. Load Balancing

Load balancing is an important issue in every distributed network. In our system, optimal performance can only be achieved if all nodes take the same share of the overall load.

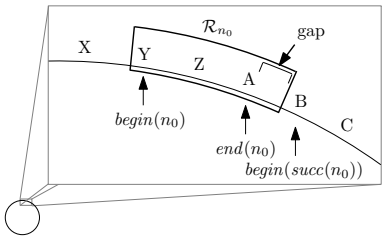
We distinguish between **storage load** and **query load**. We define the storage load to be the number of triples a node stores. The query load is the size of all messages being sent and received for query evaluation. These types of load can be totally unrelated. A node might just store a few triples but if these triples are popular it might yet suffer high query load. In our experiments, see [4], we saw that the query load and therewith the network traffic is a greater issue than storage load for the overall query throughput of the network.

Normally attribute-value pairs are stored in a DHT and the attribute, which is used to calculate the hash, is unique. If the hash function creates uniformly distributed hash values, data will be distributed uniformly, too. This natural DHT load balancing mechanism does not work with the necessary 3-times distribution of RDF triples. Triples which share a common URI at an arbitrary position are mapped to the same peer and even to the same position in DHT space. As some URIs occur more often than others (e. g. `rdf:type`), load equality can not be achieved by using the DHT load balancing mechanisms only.

We have examined different load balancing mechanisms in the BabelPeers project. **Replications** can be used to balance the query load but creates a higher storage load. We have tested two different replication approaches; static replication, where we use R different hash functions for the data distribution, and dynamic replication, where we created an overlay tree to replicate only overloaded nodes. The experiments showed that replication creates a high storage load and many message being sent over the network due to soft-state updates, while giving little benefit regarding the load imbalance. The **relocation** of triples in order to relieve overloaded nodes is a next step to avoid such additional costs. An overloaded node partitions all or just a part of its triples to one or several child nodes. A query has to be forwarded to the child and the query node combines all answers to one complete answer. The children nodes can partition their knowledge recursively as well, if they are still overloaded with their part. Detailed descriptions of these approaches and the corresponding tests and results are published in [4] and [5].

In this chapter we want to introduce a novel idea how a Chord like p2p network can be used to store RDF triples but without using a distributed hash table. We call this the **Distributed Range Ring** (DRR) and it uses a lexicographical order of triples for lookups. The peers define their responsibility ranges dynamically based on the current load situation in the network. We will show that the query algorithms presented in the previous section work seamlessly with the new balancing strategy.

The new routing scheme is based on top of the existing DHT network which is used only to maintain basic properties such as connectivity. Each triple is stored 6 times, according to each possible reordering of the three elements subject (S), predicate (P), and object (O). Thus we have the sorting orders SPO, SOP, PSO, POS, OSP, and OPS. We call



(a) Range of node n_0

```
input Target Instance i
input Routing Table begin(finger(n + 2^x))
foreach j ← log(G) − 1, ..., 0 {
  from ← begin(finger(n + 2^j));
  to ← begin(finger(n + 2^(j+1)));
  if (from ⪯ to and from ⪯ i ≺ to)
       or (from ≻ to and from ⪯ i ∨ i ≺ to) then {
    route message to node n + 2^j;
  return;
  }
}
/* message to myself */
```

(b) Routing Algorithm

Figure 8. Dynamic Range Ring

each of these variants together with the sorting order an *instance* of the triple, written as $\langle p_1, p_2, p_3, p_4 \rangle$, where p_1 is the sorting order and p_2, p_3, p_4 are the three components of the triple according to the sorting order. The identification of this sorting order is used as the first sorting criteria. p_2 through p_4 follow in this order as tie-breakers. Thus we have a well defined total relation $\prec$ on the set of instances. $\preceq, \succ, \succeq$ are defined analogously. For every instance i, $nil := \langle nil, nil, nil, nil \rangle \preceq i$

Each node stores an instance which marks the beginning of its responsibility range, denoted by $begin(n)$. Normally, this value is smaller than the beginning of the responsibility range of its successor, i. e. $begin(n) \prec begin(succ(n))$. However, there is exactly one node n_0 in the network where $begin(n_0) \succeq begin(succ(n_0))$. So the responsibility range of a node n, denoted by $\mathcal{R}_n$, is defined as follows:

$$\mathcal{R}_n := \begin{cases} \{i \mid begin(n_0) \preceq i \vee i \prec begin(succ(n_0))\} & \text{for node } n_0 \\ \{i \mid begin(n) \preceq i \wedge i \prec begin(succ(n))\} & \text{others} \end{cases}$$

From this definition it is clear that $begin(n_0)$ is not necessarily the smallest triple stored on node n_0. It rather marks the beginning of the responsibility range. Figure 8(a) shows this graphically.

Each instance i which is stored on some node n must be in this range: $i \in \mathcal{R}_n$. The set of instances which are actually stored on the node is denoted by $\mathcal{S}_n$. Thus $\mathcal{S}_n \subset \mathcal{R}_n$.

Additionally, we have the notion of the end of the responsibility range for each node. For nodes other than n_0, this is the largest instance stored on the node due according the $\prec$ relation. However, for n_0, it is the largest instance which is smaller than the beginning of the responsibility range for the successor node.

In general, this means that neither $begin(n)$ nor $end(n)$ must correspond to actually existing instances. Furthermore, there is always a gap between $end(n)$ and $begin(succ(n))$. However, new instances which are stored in the network and belong to such a gap will be stored on node n, and will change the value $end(n)$. Initially, each node is assigned a $begin(n)$ value such that there is no triple in the network within the new responsibility range of the node. Thus a node n can join between arbitrary existing nodes. Assume it joins between node n_1 and $succ(n_1)$. There is always an instance i which is within the gap between the two nodes, i. e. $end(n_1) \prec i \prec begin(succ(n_1))$. Any of these instances i will be the new value for $begin(n)$.

For the routing, each node stores the $begin(n)$ values of all nodes in its finger table, which is given by the underlying DHT network. These are typically located at the distances $2, 4, 8, \ldots$ from the current node.

We follow the same routing strategy as Chord. In each step, we try to skip as many nodes on the ring as possible in clockwise direction, without passing the target node. This leads to the routing algorithm shown in figure 8(b). Here G denotes the size of the identifier space of the DHT network. With this algorithm we reach the target node within $O(\log(N))$ routing steps with high probability.

The load balancing strategy in the DRR network is as follows. If a node discovers that it carries too much load, it will try to reduce its load by moving some triples to its successor. To avoid unnecessary balancing operation, each node has a tolerance threshold. The current load information can be used to calculate the overload factor, and if this is greater than the threshold, balancing will be triggered.

Imagine that node n_{ov} is overloaded. The number of triples to move depends on the detected overload. These triples are the largest triples (according to $\prec$) of n_{ov}, denoted by $\mathit{diff}(n_{ov})$. To fulfill the invariants of the network, n_{ov} shifts the $\mathit{diff}(n_{ov})$ triples to $succ(n_{ov})$. After that, $begin(succ(n_{ov}))$ is shifted counter-clockwise to reflect the leftmost triple in $\mathit{diff}(n_{ov})$. All nodes, whose routing tables include $succ(n_{ov})$ will learn the new value through a stabilization protocol which regularly updates the routing tables.

Meanwhile, the routing will still be correct. Until the routing tables have been updated, requests directed to the shifted triples will still be routed to n_{ov}. This node knows the correct value for $begin(succ(n_{ov}))$ and forwards the request.

The query evaluator works analogously to the BabelPeers evaluator described above except for minor modifications. In the DRR, the individual pattern lookups are now seen as range queries. To define the range, the node processes all available query information and generates a virtual instance to route the query. E.g. a pattern like `(A,D,?v1)` becomes a range query with the range $\langle \texttt{SPO}, \texttt{A}, \texttt{D}, \bot \rangle$ to $\langle \texttt{SPO}, \texttt{A}, \texttt{D}, \top \rangle$, where $\bot$ and $\top$ denote the smallest and greatest possible value, respectively. The query is routed to $\langle \texttt{SPO}, \texttt{A}, \texttt{D}, \bot \rangle$ and forwarded in clockwise order until $begin(succ(n))$ is bigger than the end of the range or the node n_0 is reached. The query issuing node combines the one or more answers which make up the complete result. Thus, every possible graph pattern can also be evaluated with DRR.

3.4. WS-MDS Interface

As mentioned already, our goal is to create a flexible and scalable resource description service. To show that BabelPeers fulfills these requirements, we implemented a prototype discovery service based on a BabelPeers backend for the Globus Toolkit, which can replace the original WS-MDS service.

The Globus WS-MDS implementation is based on a hierarchical model, where data sources are located at the leaves of a tree (or DAG). Each data source publishes its data to index services and the index services propagate and aggregate the information upwards in the hierarchy. This means either that an index service at the root of the hierarchy has to cope with high data and query load or that data needs to be aggregated and therefore information is lost.

A DHT based p2p model without a distinguished root-node provides a global view without aggregation and without a single point of entry/failure. Thus, we think it is beneficial to use BabelPeers within Globus-based Grids.

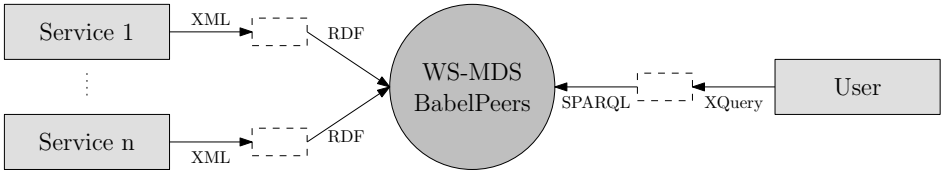


Figure 9. WS-MDS with BabelPeers backend.

However, there is a gap between the WS-MDS XML and the RDF data model. To provide compatibility with existing services, which publish their properties as XML documents, we employ a generic XML to RDF translation. Unfortunately, querying the resulting documents is complicated, as they have an artificial structure. Thus we favor a more natural way of using RDF as described in section 2. Although it requires more work to write appropriate translators, it can provide powerful and flexible resource queries. The same holds for the query language. There is no one-to-one mapping of XPath to the RDF query language SPARQL used in BabelPeers. We tested an automated partial translation, but again we favor using native SPARQL to unleash the full power of the system. The architecture is shown in figure 9. The dashed boxed indicate translators that have to be implemented individually.

4. Performance Evaluation

In order to evaluate the algorithms and to compare the different versions, we have implemented BabelPeers prototypically. The prototype can be used both for measurements and simulations. Real-life measurements give a better insight into the real behavior of the system; however, they are limited in the number of available nodes. We use a compute cluster with up to 128 nodes for our experiments. Simulations complement the experiments by giving results also for networks with more than 128 nodes. In this section, we present a selection of the performed experiments with a focus on query evaluation performance and load balancing.

4.1. Generation of Test Data

We have created test data reflecting the expected structure of RDF-based information in our scenario. The goal of the test data is to evaluate all aspects of the system. Thus we designed a flexible way to generate test data that includes complex schema information and queries that result in various access patterns.

We generate the test data in multiple steps. We start by generating a **class hierarchy** and a **property hierarchy**. These hierarchies are generated using the same algorithm, however with different parameters. The algorithm outputs a directed acyclic graph (DAG), whose elements are either classes or properties. The parameters determine how tree-like the DAG will be, which means up to what degree multiple inheritance is used. They further determine how deep the hierarchy is, i. e. how long the average path from an element to the top element is. Using the class and property hierarchies, we generate instances and relations between these instances. The instances are typed over the classes and the relations between the individuals are generated using the properties from the second hierarchy. For the experiments presented here, we generated two test-sets, one

with approx. 110k triples, and one with approx. 1.1 Mio triples. RDFS reasoning approx. doubles the number of triples.

The queries we use are generated as follows: We start by selecting a random subgraph of the current test set. This ensures that each query will have at least one match. In order to have queries of varying difficulty, we randomly replace URIrefs that occur in the subgraph by variables. During this process, we ensure a unique variable is used for each replaced URIref; however, we do not always replace all occurrences of a URIref. This way, we generated 1000 queries for each test set.

4.2. Query Performance

To evaluate the query performance, we start both a p2p network node and a query client on each physical node of the cluster partition used for the experiment. The client loops permanently sending a batch of test queries. The whole network runs for an hour, and we calculate the overall throughput by counting how many queries are evaluated correctly during this hour. We used an increasing number of nodes from 1, 2, 4, up to 128 nodes of our cluster. The model graph of the test set is split up into N chunks, where N is the current network size, and one chunk is assigned to each node. Thus each triple is stored initially only on one node, and then disseminated to the network. Furthermore, the RDFS reasoning is switched on so that new triples are generated during the run of the experiment.

For the evaluation of the results, we compute locally the answers after all RDFS reasoning. We compare the results of the experiments with these results and count only those results that are correct. Thus a query that returns only partial results is not counted in the throughput. This happens during the startup phase, when RDF reasoning is still in process.

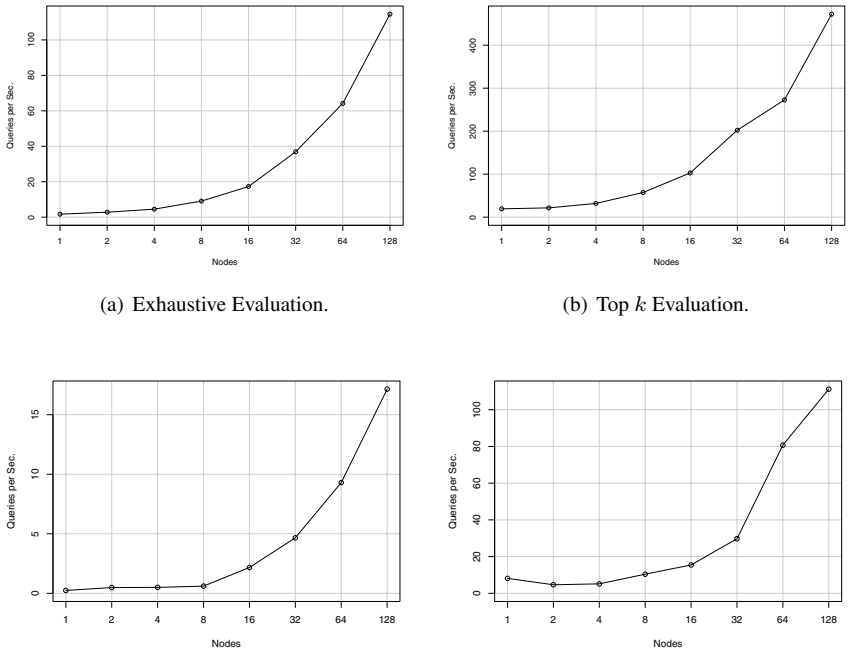
The result is shown in figure 10(a). Overall, we can see that the scalability of the system is very good. The top k evaluation generates a much higher throughput. This is clear as fewer results are generated, tapping smaller amounts of data in the network. The results are shown in figure 10(b). Here, the speedup for 2 nodes is relatively small (1.12). As the top k method needs less CPU time for the subgraph matching, because fewer hits are searched, overall the network communication cost has a greater relative impact on the performance.

For larger networks, the figures get better. For example, we observe that the speedup increases by a factor of 1.73 between 64 nodes and 128 nodes.

The results for the larger testset are shown in figure 10(c) and figure 10(d). The overall throughput is smaller. This is partly because much more data has to be transferred. However, another reason is that some of the queries now have huge result sets which have to be generated. Both effects are smaller in the top k evaluation. Another effect is that the for small network sizes, we do not experience good speedup values. This is again due to the dominance of the data transfers. However, for larger networks, the relative gain in performance is good.

4.3. Triple Dissemination and Reasoning

We now look at the time it takes to disseminate the triples and to do the RDF Schema reasoning. For this, we started a new blank network and evaluated how long it takes until



(a) Exhaustive Evaluation.

(b) Top k Evaluation.

(c) Exhaustive Evaluation, 1 Mio.

(d) Top k Evaluation, 1 Mio.

Figure 10. Overall Throughput of the Prototype.

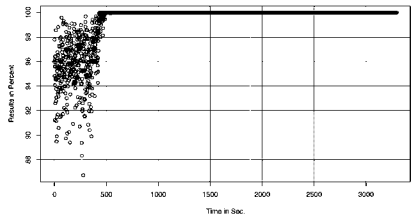


Figure 11. Duration of the RDF Schema Reasoning Process.

the first queries are answered correctly. Look at figure 11. It shows which percentage of the queries are answered correctly on average during every second after the startup of a new network. In this percentage, we also counted partial results. We can see, that during the first ten minutes, on average between 86 and 100 percent of the expected results have been delivered. After that phase, all conclusions were fully generated, thus every query had been answered correctly.

4.4. Load Balancing

Experiments about the replication and relocation balancing strategies can be found in [4,5]. This section focuses on load balancing experiments according to the Dynamic

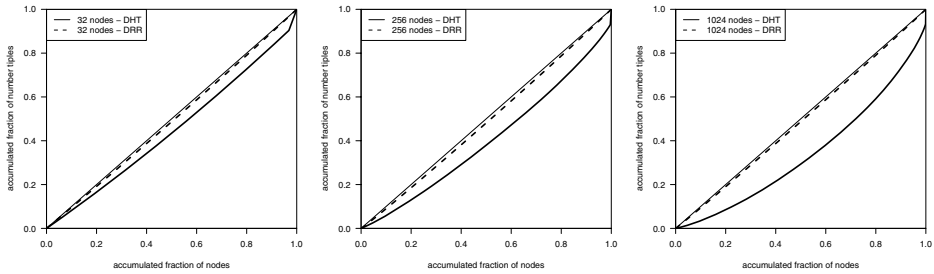


Figure 12. Lorenz curves for triple distribution

Range Ring. In our evaluations, we examine load balancing regarding to the two different kinds of load, **storage load** and **query load**. All experiments are done as simulations using the 110k triples testset.

Figure 12 depicts Lorenz-curves for the triple distribution on networks with sizes of 32, 256, and 1024 nodes. In the case of the DHT network, no loadbalancing has been done, in the case of DRR, load was balanced by **storage load**. The nodes are sorted by the number of stored triples in ascending order. Each curve shows, how many percent of all triples the first $p\%$ nodes account for, or the other way around, how many percent of all nodes are responsible for the first $p\%$ of all triples. For example the first diagram (32 nodes - DHT) shows that the first 95% of all nodes store about 85% of all triples.

For an increasing number of nodes in a standard DHT network for RDF data, the curves bend more and more. This means that a higher percentage of nodes store only few triples and the storage load imbalance increases. The new DRR does not show such a behavior. The distribution of the RDF triples is indifferent to the size of the network.

The second and more important strategy balances based on the **query load**. For each triple we count the number of times it was part of an answer message and then shift triples in the network such that the sum of the count of delivered triples is approximately equal for all nodes.

We think that the query profile, the set of queries posed to the network, is rather stable and does not change very quickly, as topics of interest change slowly; e. g. queries for a very modern cluster architecture. So the DRR can adjust itself to the current query profile and in the case of a slowly changing profile, just a few balancing operations are needed to adapt itself to new load situations.

For measuring the performance of this load-balancing strategy, we generated 30 query sequences QS_i by randomly choosing $n = 10.000$ queries from the whole set of 1.000 queries. This means that the number of times each query is posed in a query sequence is normally distributed around $n \cdot p = 10$ times with standard deviation $\sigma = \sqrt{n \cdot p \cdot (1-p)} \approx 3.16$. This still allows for quite some variance between the QS_i.

Now we inserted all triples into the network and balanced the nodes by storage load. Based on this distribution we processed each query sequence QS_i and recorded for each QS_i for each node how many triples it sent over the network for query processing. These values, one per node, are collected into the set Q_i (query load for query sequence i).

In the next step we posed the query sequences QS_i again. Before processing the query sequence QS_i, however, we performed a load balancing operation by the query load of the previous query sequence Q_{i-1}. Hence, for each query sequence the nodes created query load (for sending triples as answers to queries) and load-balancing load

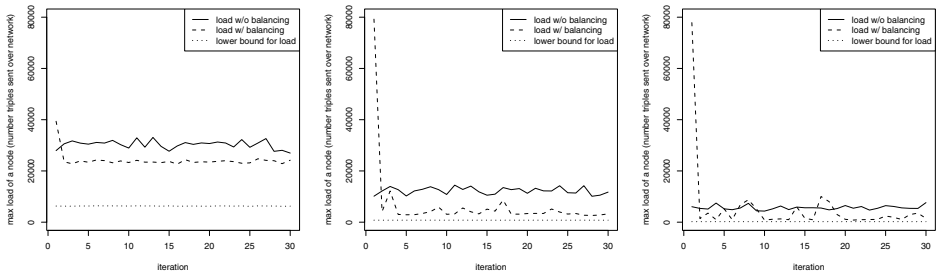


Figure 13. Load in the network

(for sending triples to their neighbors). The sum of query load and load-balancing load of all nodes in iteration i is denoted as the set QLB_i.

Our goal is now to compare the maximum load caused by processing queries and balancing the load for any node ($\max QLB_i$) with the maximum load caused by the queries for any node if no balancing occurs ($\max Q_i$). Furthermore, we want to compare these figures to the maximum load of a node caused by query processing in case the triples were optimally placed. We can give a lower bound for the latter as $OPT_i = \sum_i Q_i / \text{number\_nodes}$. In this case the total load for query sequence Q_i is evenly distributed among all nodes.

Figure 13 shows the values of $\max QLB_i$, $\max Q_i$, and OPT_i for 30 iterations with network sizes of 32, 256, and 1024 nodes and a tolerance threshold of 100% (see section 3.3). Note that the query load without balancing is measured after an initial round of load balancing by storage load was performed. In a regular DHT (as opposed to the DRR) these figures would be therefore worse.

We see that for the 32 node network, the tolerance level of 100% seems to be a little bit too high. The query load imbalance is not very strong and only few nodes triggered balancing operations. A closer look to the simulation logs shows that only during the first iteration 6 nodes performed load balancing and then no further balancing operations were done.

If we increase the network size to 256 nodes, the imbalance grows. The relative gains increase and the load comes pretty close to the best load possible (perfect distribution of query load). We see a typical development that the first balancing operation is pretty costly. After that, however, the necessary load balancing operations are very cheap, in case the query profile remains rather similar, and the query evaluation benefits greatly from the good load balance.

Finally, at the simulated network of 1024 nodes, we see that a low tolerance setting sometimes created a large overhead due to load balancing operations. Decreasing the tolerance even further increased the overhead as well. This is caused by waves going through the network. Suppose a network is almost perfectly balanced but one node has too much load. This load needs to be shifted over long distances because the node's immediate neighbors might have perfect load and thus do not want to take additional triples. By increasing the number of nodes in the network, the average query load per node decreases. The reason for this is that the total work to be done remains constant even if the number of nodes in the network increases. The work is only defined by the number of queries per iteration. So some of the balancing operations are expensive while the query load of nodes is small as they store just few triples.

Several strategies can be applied to reach better results in this case. First, the tolerance setting can be increased. Second, the frequency of load-balancing operations can be decreased. In this case the ratio of load-balancing overhead compared to the query load decreases and so it carries not such a weight. Finally, other strategies such as migrating nodes can be applied to move less loaded nodes to positions of higher loaded nodes.

We can conclude that, with correct settings, the DRR can create huge improvements to the problem of load-imbalance.

5. Conclusion

In this chapter, we have described our BabelPeers resource discovery system. The system combines both semantic features for advanced description and querying of resources, and efficient and scalable query evaluation. BabelPeers uses RDF to describe the resources, performs reasoning to generate new information, and integrates resource descriptions stored at different nodes. The triples are disseminated using a DHT based P2P network.

We have explained two different query evaluation strategies. The first one aims to exhaustively collect every answer to a given query, while the second one only retrieves the top k matches with respect to a user-defined attribute. As load-balancing is an important issue in DHT based systems, we have additionally presented a strategy that efficiently balances either the query load or the storage load of the network by local interactions.

We have evaluated the system by doing benchmarks with up to 128 nodes, complemented by simulations. The benchmarks show that the query evaluation is fast and scales well to larger numbers of nodes. The top k query evaluation has a much higher throughput than the exhaustive strategy, at the price of not retrieving every match.

To show the usability of the system in real life, we have further developed a wrapper that exposes the system's functionality via a Globus WS-MDS interface. Using this wrapper the system can be used as a replacement for the original MDS component, thereby exchanging the hierarchical nature of the MDS architecture with a global view including all resources even of large Grid systems without performance penalties.

Acknowledgement: Partially supported by the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS).

References

[1] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim van Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In *International Semantic Web Conference (ISWC)*, volume 3298 of *LNCS*, pages 107–121, 2004.

[2] George Anadiotis, Spyros Kotoulas, and Ronny Siebes. An architecture for peer-to-peer reasoning. In *New forms of reasoning for the Semantic Web: scalable, tolerant and dynamic @ International Semantic Web Conference '07*, 2007.

[3] Sergio Andreozzi, Stephen Burke, Laurence Field, Steve Fisher, Balazs Kónya, Marco Mambelli, Jennifer M. Schopf, Matt Viljoen, and Antony Wilson. Glue schema specification, version 1.2, final specification. `http://infnforge.cnaf.infn.it/glueinfomodel/uploads/Spec/GLUEInfoModel_1_2_final.pdf`, December 2005.

[4] Dominic Battré, Felix Heine, André Höing, and Odej Kao. Load-balancing in P2P based RDF stores. In *Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*, Athens, Georgia, USA, November 2006.

[5] Dominic Battré, Felix Heine, André Höing, and Odej Kao. On Triple Dissemination, Forward-Chaining, and Load Balancing in DHT based RDF stores. In *Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2006)*, Seoul, Korea, September 2006.

[6] Dave Beckett. RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar`, 2004.

[7] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.

[8] Dan Brickley and Ramanathan V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. `http://www.w3.org/TR/rdf-schema`, 2004.

[9] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 54–68, London, UK, 2002. Springer-Verlag.

[10] John Byers, Jeffrey Considine, and Michael Mitzenmacher. Simple Load Balancing for Distributed Hash Tables. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS 03)*, pages 31–35. Springer, 2003.

[11] Min Cai, Martin Frank, Baoshi Pan, and Robert MacGregor. A Subscribable Peer-to-Peer RDF Repository for Distributed Metadata Management. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 2(2):109–130, 2004.

[12] Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. Grid Information Services for Distributed Resource Sharing. In *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001.

[13] Prasanna Ganesan, Mayank Bawa, and Hector Garcia-Molina. Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. In *VLDB '04: Proceedings of the 30st international conference on Very large data bases*, pages 444–455, 2004.

[14] Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties. In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[15] Patrick Hayes. RDF Semantics. `http://www.w3.org/TR/rdf-mt`, 2004.

[16] Felix Heine. Scalable P2P based RDF Querying. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 17, New York, NY, USA, 2006. ACM Press.

[17] Erietta Liarou, Stratos Idreos, and Manolis Koubarakis. Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors, *The Semantic Web – ISWC 2006*, volume 4273 of *LNCS*, pages 399–413, November 2006.

[18] Frank Manola and Eric Miller. RDF Primer. `http://www.w3.org/TR/rdf-primer`, 2004.

[19] Akiyoshi Matono, Said Mirza Pahlevi, and Isao Kojima. RDFCube: A P2P-based Three-dimensional Index for Structural Joins on Distributed Triple Stores. In *Proceedings of Fourth International Workshop on Database, Information Systems and Peer-to-Peer Computing (DBISP2P 2006)*, 2006.

[20] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA: a P2P networking infrastructure based on RDF. In *WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA*, pages 604–615, 2002.

[21] Rajesh Raman, Miron Livny, and Marvin Solomon. Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching. In *Proceedings of the Twelfth IEEE International Symposium on High-Performance Distributed Computing*, Seattle, WA, June 2003.

[22] Rajesh Raman, Miron Livny, and Marvin H. Solomon. Matchmaking: Distributed Resource Management for High Throughput Computing. In *HPDC*, pages 140–, 1998.

[23] Domenico Talia, Paolo Trunfio, and Jingdi Zeng. Peer-to-Peer Models and Services for Resource Discovery on Large-scale Grids. In *VECPAR'06*, 2006.

[24] Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, and Dave Reynolds. Efficient RDF Storage and Retrieval in Jena2. In *Proceedings of SWDB'03, The first International Workshop on Semantic Web and Databases*, pages 131–150, 2003.

[25] Yingwu Zhu and Yiming Hu. Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems. *IEEE Transactions on Parallel and Distributed Systems*, 16(4):349–361, 2005.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Data Integration based on Schema-Mapping in Service-based Grids

Carmela COMITO [a,1] and Domenico TALIA [a]
[a] *University of Calabria, Via P. Bucci 41 c, 87036 Rende, Italy*

Abstract. Data integration is a key issue for exploiting the availability of large, heterogeneous, distributed and highly dynamic data volumes on Grids. In the Grid, a centralized structure for coordinating all the nodes may not be efficient because it becomes a bottleneck when a large number of nodes are involved and, most of all, it does not benefit from the dynamic and distributed nature of Grid resources. In this chapter, we present a decentralized service-based data integration architecture for Grid databases that we refer to as *Grid Data Integration System* (GDIS). GDIS main concern is reconciliation of schema heterogeneity among data sources. GDIS adopts the decentralized XMAP integration approach that can effectively exploit the available Grid resources and their dynamic allocation. The GDIS architecture offers a set of Grid services that allow users to submit queries over a single database and receive the results from multiple databases that are semantically correlated with the former one. The underlying model of such architecture is discussed and its implementation based on the OGSA Globus architecture is described. Moreover, we show how it fits the XMAP formalism/algorithm.

Keywords. Data Grids, Grid Services, XML Schema-mapping

Introduction

A Grid [4] can include related data resources maintained in different syntaxes, managed by different software systems, and accessible through different protocols and interfaces. Due to this diversity in data resources, one of the most demanding issues in managing data in Grids is reconciliation of data heterogeneity. Therefore, in order to provide facilities for addressing requests over multiple heterogeneous data sources, it is necessary to provide data integration models and mechanisms.

The Grid community is devoting great attention toward the management of structured and semi-structured data such as relational and XML data. Two significant examples of such efforts are the *OGSA Data Access and Integration* (OGSA-DAI) [9] and the *OGSA Distributed Query Processor* (OGSA-DQP) [10] projects. However, there is a lack of middleware schema-integration services. To the best of our knowledge, there are only few works designed to provide schema-integration in Grids [11, 12].

The large-scale, highly dynamic and autonomous nature of data sources in Grid raises new challenges in data integration systems and traditional approaches to data in-

[1] Corresponding Author: Carmela Comito, DEIS, University of Calabria, Via P. Bucci 41 c, 87036 Rende, Italy ; E-mail: ccomito@deis.unical.it.

tegration, such as hierarchical mediator/wrapper-based systems [6], are not suitable in Grid settings. Further, the distributed nature of the Grid proves that it is not suitable to a centralized structure that coordinates the activity of all of the nodes in the system mostly because it can become a bottleneck. A decentralized approach can effectively exploit the available Grid resources and their dynamic allocation according to the trend in the database community where the P2P paradigm has been adopted to overcome the limitations of distributed database systems and to exploit the dissemination of data sources over the Internet. According to these trends, we developed a framework, XMAP [13], for integrating heterogeneous XML data sources distributed over a large-scale, highly dynamic network of autonomous nodes. We propose a query reformulation algorithm to combine and query XML sources through a decentralized point-to-point mediation process among the different data sources by using P2P schema mappings. The proposed integration model is based on path-to-path mappings, using the XPath language.

We exposed the above cited XML integration formalism as a Grid Service within an OGSA-based Grid architecture, the *Grid Data Integration System* (GDIS) [15]. GDIS is a service-based architecture for providing data integration in Grids using a decentralized approach. The GDIS infrastructure exploits the middleware provided by OGSA-DQP, OGSA-DAI, and Globus Toolkit [3] introducing on top of these building blocks schema-integration services by wrapping as a Grid service the XMAP integration approach. More precisely, the proposed GDIS system offers a decentralized wrapper/mediator-based approach to integrate data sources: it adopts the XMAP decentralized mediator approach to handle semantic heterogeneity over data sources, whereas syntactic heterogeneity is hidden behind OGSA-DAI wrappers. So, by deploying the XMAP framework in OGSA-Grids, we combine the decentralized, flexible and scalable XMAP mediation approach with Grid infrastructure and services allowing in such a way wide-scale, flexible, reliable semantically enriched data sharing: in the GDIS systems when a query is posed using a node schema, answers should come from anywhere in the system thanks to the XMAP semantic connections.

Among the few works designed to provide schema-integration in Grids, the most notable ones are *Hyper* [11] and *GDMS* [12]. Both systems are based on the same approach that we have used ourselves: building data integration services by extending the reference implementation of OGSA-DAI. The *Grid Data Mediation Service* (GDMS) uses a wrapper/mediator approach based on a global schema. GDMS presents heterogeneous, distributed data sources as one logical virtual data source in the form of an OGSA-DAI service. This work is essentially different from ours as it uses a global schema. For its part, *Hyper* is a framework that integrates relational data in P2P systems built on Grid infrastructures. As in other P2P integration systems, the integration is achieved without using any hierarchical structure for establishing mappings among the autonomous peers. In that framework, the authors use a simple relational language for expressing both the schemas and the mappings. By comparison, our integration model follows as Hyper an approach not based on a hierarchical structure, however differently form Hyper it focuses on XML data sources and is based on schema-mappings that associate paths in different schemas.

The remainder of the chapter is organized as follows. Section 1 describes the XMAP integration formalism and the XPath reformulation algorithm. Section 2 presents the Grid Data Integration System describing the logical model and the the system architecture implementation by using OGSA services. Section 3 presents the OGSA-based XML Data

Integration Service illustrating its components and its internal behavior by outlining the typical service interactions in the system. Preliminary experimental results of the GDIS prototype are illustrated in section 4. Finally, section 5 draws some conclusions.

1. The XMAP Integration Formalism

The XMAP framework [13], semantically relates XML schemas enabling the formulation of queries over heterogeneous, distributed XML data sources. The environment is modeled as a system composed of a number of Grid nodes, where each node can hold one or more XML databases. These nodes are connected to each other through declarative mappings rules.

1.1. Integration Model

As mentioned before, a traditional centralized architecture of data integration systems is not suitable for highly dynamic and distributed environments such as the Grid. Thus, we designed an approach inspired from [8] where the mapping rules are established directly among source schemas without relying on a central mediator or a hierarchy of mediators. Therefore, in our integration model, there is no global schema representing all data sources in a unique data model but a collection of local schemas (the native schema of each data source). Regardless of the total number of nodes composing the system, each source schema is directly connected to only a small number of other schemas. However, it remains reachable from all other schemas that belong to its "transitive closure". For any mapping M, its closure is defined as the set of rules that can be derived from M by repeated composition of schema paths. In other words, the system supports two different kinds of mapping to connect schemas semantically: *point-to-point* mappings and *transitive* mappings. In transitive mappings, data sources are related through one or more "mediator schemas". For example, if we have a source A directly connected to a source B and B connected to C, A is connected to *both* B and C. Establishing the mappings this way creates a graph of semantically related sources where each of the sources knows its direct semantic neighbors (point-to-point mapping) and can learn about the mappings of its neighbors (transitive mapping). Therefore, in our integration model all nodes are equal: there is no distinction between data sources and mediators. Each node acts both as a data source contributing data and as a local mediator providing an uniform view over the data provided by other nodes.

Our integration model is based on schema mappings to translate queries between different schemas. The goal of a schema mapping is to capture structural as well as terminological correspondences between schemas. We address this goal by associating paths in different schemas. Mappings are specified as path expressions that relate a specific element or attribute (together with its path) in the source schema to related elements or attributes in the destination schema. The data integration model we designed is indeed based on path-to-path mappings expressed in the XPath [18] query language, assuming XML Schema as the data model for XML sources. Specifically, this means that a path in a source is described in terms of XPath expressions.

The mapping rules are specified in XML documents called XMAP documents (see Figures 1, 2). Each source schema in the framework is associated to an XMAP document containing all the mapping rules related to it.

1.2. The XMAP Reformulation Algorithm: a Case Study

Our query processing approach exploits the semantic connections established in the system by performing the *XPath query reformulation algorithm* before executing the input query, in order to gain further knowledge. This way, when a query is posed over the schema of a source, the system will be able to use data from any source that is transitively connected by semantic mappings. Indeed, it will reformulate the given query expanding and translating it into appropriate queries for each semantically related source. Thus, a user can retrieve data from all the related sources in the system by simply submitting a single XPath query.

The rationale of the algorithm is to perform single reformulation steps. A reformulation step corresponds to the reformulation of a given query only with respect to the schemas directly connected to it. So, the algorithm is composed of several reformulation steps, but each of such steps performs only direct reformulations by using the point-to-point mappings. Each time a reformulated query is obtained, the algorithm tries to rewrite it by recursively invoking the reformulation function and using its direct mappings.

The query reformulation algorithm uses as input an XPath query and the mappings, and it produces as output zero, one or more reformulated queries.

In the following we briefly describe an example of use of the XMAP algorithm. In the example we consider the information system of a travel agency composed of several, geographically distributed branches each of which holds its own XML database conforming to a specific schema. Let suppose a user wants to find the *location* of the trips made in the *year* 2000. To this aim the following query Q_{S_1} is formulated over the schema S_1:

Q_{S_1}=`/S1/trip[year = "2000"]/route/location`

In the first step the algorithm identifies the paths in the query:

P_1=`/S1/trip/route/location`
P_2=`/S1/trip/year`

Next, exploiting the XMAP document associated to the schema S_1 (see Figure 1), the algorithm finds two mapping rules connecting $S1$ to $S2$ through the paths P_1 and P_2.

More precisely, one of these rules relates P1 to two paths in $S2$, respectively `/S2/journey/itinerary/destination` and `/S2/flight/route/destination`. Similarly, the other mapping rule relates P_2 to the path `/S2/journey/year` and the path `/S2/flight/year`. In the considered example as the schema $S2$ has correspondences for both paths P_1 and P_2, it can be used to reformulate the query Q_{S_1}. In particular, the algorithm produces two direct reformulations of the query Q_{S_1} over the schema $S2$, respectively Q_{S2_1} and Q_{S2_2}.

Q_{S2_1}=`/S2/journey[year="2000"]/itinerary/destination`
Q_{S2_2}=`/S2/flight[year="2000"]/route/destination`

Since in the XMAP document associated to the schema $S1$ there are no more mapping rules involving the paths in the query Q_{S_1}, no further reformulations are produced. Then the algorithm is recursively invoked over the direct reformulations Q_{S2_1} and Q_{S2_2}, exploiting the XMAP document associated to the schema $S2$ (see Figure

```
<sourceSchema>S1</sourceSchema>
<Rule cardinality="Mapping1-N">
 <destinationSchema>S2</destinationSchema>
 <sourcePath>/S1/trip/route/location</sourcePath>
 <destinationPath>/S2/journey/itinerary/destination</destinationPath>
 <destinationPath>/S2/flight/route/destination</destinationPath>
</Rule>
<Rule cardinality="Mapping1-N">
 <destinationSchema>S2</destinationSchema>
 <sourcePath>/S1/trip/year</sourcePath>
 <destinationPath>/S2/journey/year</destinationPath>
 <destinationPath>/S2/flight/year</destinationPath>
</Rule>
```

Figure 1. Fragment of the $S1$ XMAP document.

```
<sourceSchema>S2</sourceSchema>
<Rule cardinality="Mapping1-1">
  <destinationSchema>S3</destinationSchema>
  <sourcePath>/S2/flight/route/destination</sourcePath>
  <destinationPath>/S3/trip/itinerary/location</destinationPath>
 </Rule>
 <Rule cardinality="Mapping1-1">
  <destinationSchema>S3</destinationSchema>
  <sourcePath>/S2/flight/year</sourcePath>
  <destinationPath>/S3/trip/year</destinationPath>
 </Rule>
```

Figure 2. Fragment of the $S2$ XMAP document.

2). In so doing, the algorithm finds two mapping rules connecting the schema $S2$ to the schema $S3$ through the paths composing the query Q_{S2_2}, whereas any mapping rules involving the paths of the query Q_{S2_1} have been found. As a consequence, the execution of the XMAP reformulation algorithm over the query Q_{S2_1} does not produce any reformulated query, instead the reformulation of the query Q_{S2_2} produces the reformulation Q_{S3}:

Q_{S3}=/S3/trip[year="2000"]/itinerary/location

This first recursive invocation of the algorithm ends here producing the query Q_{S3} as a direct reformulation of the query Q_{S2_2} over the schema $S3$ (that is a transitive reformulation of the original query Q_{S1}). Thus, now we can exploit the transitive mappings of the schema $S2$, by recursively invoking the reformulation algorithm over the query Q_{S3}. As the XMAP document associated to the schema $S3$ has no mappings for the paths composing the query Q_{S3}, the algorithm ends here.

2. The Grid Data Integration System

The XMAP reformulation algorithm has been deployed in the *Grid Data Integration System* (GDIS). GDIS is a decentralized service-based data integration architecture for Grid databases [15].

The design of the GDIS framework has been guided by the goal of developing a decentralized network of semantically related schemas that enables the formulation of queries over heterogeneous data sources in Grids. The system also allows distributed queries when the target data sources are located at different Grid sites. In order to achieve this aim we expose data integration utilities as Grid Services that allow for combining or transforming data from multiple heterogeneous sources to obtain integrated or derived views of data. With our proposal we aim to achieve the following goals:

i) Develop a decentralized architecture for data integration in service-based Grids.
ii) Expose data integration utilities as Grid Services allowing to integrate different data models on distributed databases.
iii) Provide flexible integration features addressing the challenges arisen from large-scale, autonomous, dynamic data sources.
iv) Exploit the middleware provided by OGSA-DAI, OGSA-DQP and Globus Toolkit.
v) Use the facilities of the OGSA to dynamically achieve resource discovery and allocation.
vi) Develop an ease of use, open, extensible, transparent and maintainable mediation service. Users should be able to run their queries on the Grid in an easy and transparent way, without needing to know details of the Grid structure and operation, network features, physical location of data sources or their query capabilities.

2.1. GDIS Logical Model

The GDIS system is envisioned as a set of Grid nodes that accomplish one or more of the typical tasks of a data integration system. The main entities and associated tasks in GDIS are: (1) *data providers* supplying data source with or without its schema, (2) *mediators* providing only a schema to which other sources' schemas may be mapped, (3) *clients* formulating queries.

The GDIS system offers a wrapper/mediator-based approach to integrate data sources: it adopts the XMAP decentralized mediator approach to handle semantic heterogeneity over data sources, whereas syntactic heterogeneity is hidden behind ad-hoc wrappers. More precisely, the proposed framework is characterized by three core components: a *query reformulator engine*, a *distributed query processor*, and a *wrapper module* (see Figure 3).

As a consequence of these features, some nodes in the system supply appropriate services addressing the challenges characterizing the design of a wrapper/mediator-based data integration system. Thus, we can further classify the nodes in the system on the basis of the functionality they achieve with regard to the above mentioned features, obtaining in such way the following categorization:

1. *processing nodes* that supply distributed query processing capabilities aiming at compile, optimise, partition and schedule distributed query execution plans over multiple *execution nodes*; every execution node is in charge of a sub-query execution plan assigned to it by a processing node;

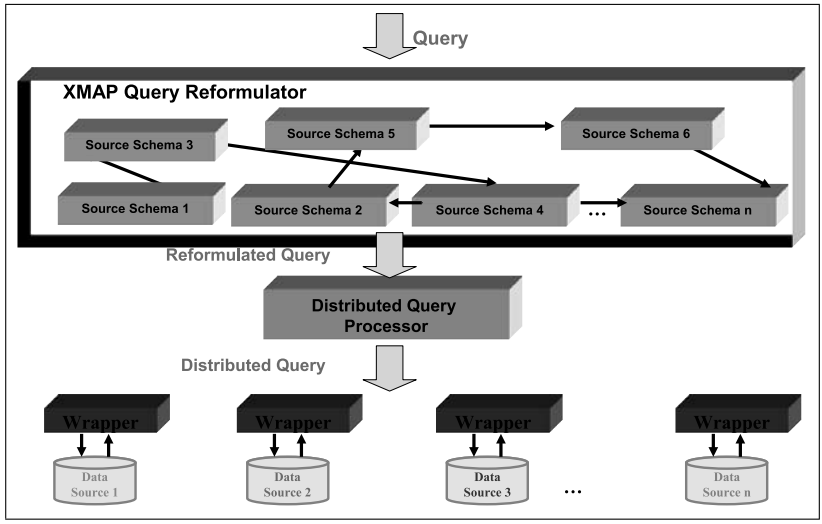


Figure 3. GDIS: a decentralized wrapper/mediator-based integration system.

2. *data integration nodes* that offer: (i) a set of data integration utilities allowing to establish mappings, and (ii) the query reformulation algorithm introduced by the XMAP integration formalism. These utilities are exposed through the portTypes of the proposed OGSA-based GDI data integration service;
3. *wrapper nodes* allowing for access actual data sources.

It should be noted that any Grid node can carry out all the mentioned services.

Below, we briefly outline the very general behavior of the proposed integration system, regardless of the specific implementation paradigm (for the detailed description of the distributed and service-oriented implementation of the GDIS integration system see sections 2.2 and 3). The user query is handled by the reformulator engine that through the XMAP query reformulation algorithm produces reformulations of the original query. All the obtained reformulations (included the original query) are then processed by the distributed query processor (DQP) that partitions each of such queries in several sub-queries. Then, each produced sub-query execution plan is processed independently by the DQP that through the wrapper accesses data source and produces the partial query result. After that, the DQP collects the sub-query results into the final query result and returns it to the querying node.

2.2. GDIS Architecture Implementation

We designed our data integration system as a service-based distributed architecture. So, the proposed architecture is general enough to be feasibly fulfilled in both the GT3 and GT4 implementations of the Globus Toolkit. At present, the system is integrated with GT3 based on OGSA; minor modifications will be required to port the system to the WSRF-based GT4 [17]. In particular, in the Globus Toolkit 4 Grid services will be replaced by WS-Resources, service data by resource properties and Index Services by ServiceGroups.

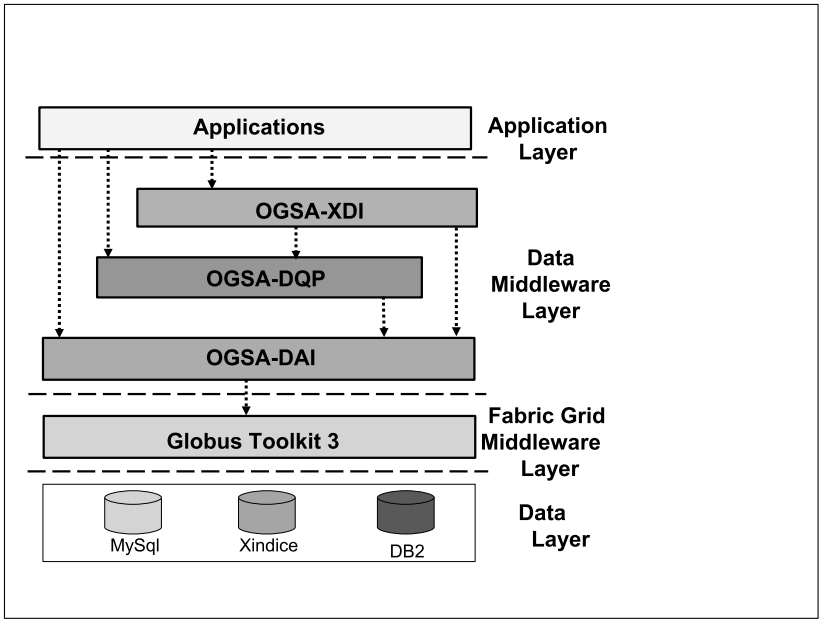


Figure 4. GDIS layered architecture.

We implemented the prototype of the Grid Data Integration System by reusing the framework provided by OGSA-DAI 5.0. The service runs in the Globus 3 framework and is hosted within Jakarta-Tomcat [16] as a container.

Each node in GDIS exposes all its resources as Grid services except data resources and data integration facility that are exposed as Grid Data Services (GDSs). In so doing, the GDIS system introduces the OGSA-based *XML Data Integration* (OGSA-XDI) service that extends OGSA-DAI with additional portTypes (see Figure 4).

As said before, the proposed framework is characterized by three core components: a query reformulation engine, a wrapper module, and a distributed query processor. The implementation choices concerning these components are the following (see Figure 5). OGSA-DAI components are employed as wrappers over databases (wrapper nodes). OGSA-DQP is applied as distributed query processor. More precisely, execution nodes keep the OGSA-DQP Grid Query Evaluation Service (GQES), whereas processing nodes expose the Grid Distributed Query Service (GDQS) also made available by OGSA-DQP. As already explained, the XMAP integration formalism is used to handle data heterogeneity. Therefore, the query reformulator engine implements the XMAP query reformulation algorithm exposed, as the other integration facilities, through the proposed OGSA-XDI service, as described in the following section. These integration facilities are provided by data integration nodes.

Note that in the GDIS system the XMAP algorithm has been integrated in OGSA-DAI by simply introducing a new activity (as it will be detailed in the following sections) whereas OGSA-DQP has been used as a stand-alone system. In another piece of work, we have embedded the XMAP algorithm within the OGSA-DQP prototype [1,2]. In this case XMAP has been wrapped as a web service that is called from within OGSA-DQP that has been properly extended.

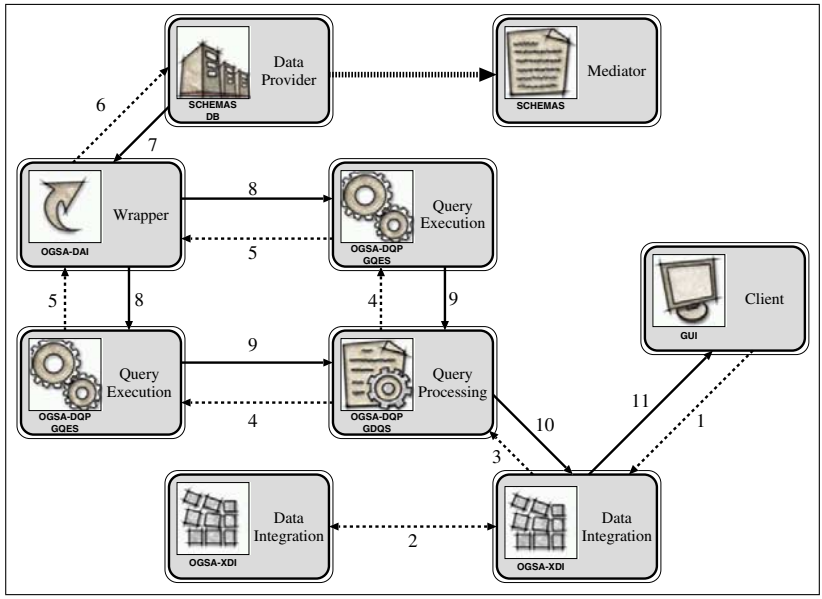


Figure 5. GDIS implementation architecture.

3. OGSA-XDI: an OGSA-based XML Data Integration Service

The GDIS system introduces the OGSA-based XML Data Integration (*OGSA-XDI)* service that extends OGSA-DAI portTypes with additional functionality to both enable users to specify semantic mappings among a set of data sources (in the form of XMAP documents) and to execute the described XMAP query rewriting algorithm.

The architecture of OGSA-DAI and the core engine are designed for new activities in order to provide new functionalities. Indeed, in order to implement the OGSA-XDI service we decided to extend the free available OGSA-DAI Grid Data Service (GDS) reference implementation. By this, we are avoiding to build (i) new proprietary solutions and (ii) reimplementing well solved aspects of Grid data services, and are able to concentrate on the integration task. According to this, we introduced a new activity, the `XPathQueryReformulation` activity, that wraps the XPath query reformulation algorithm of the XMAP framework.

The OGSA-XDI architecture is composed, analogously to OGSA-DAI, of a number of new services including:

1. *Grid Data Integration service* (GDI). This service provides schema mapping utilities for semantically connect XML data sources. To this aim the GDI extends the OGSA-DAI GDS by introducing a new activity, the `XPathReformulation` activity, devoted to the reformulation of an XPath query by using the XMAP reformulation algorithm.
2. *Grid Data Integration Factory* (GDIF) which creates a GDI on request. A GDIF extends a Grid Data Service Factory (GDSF) by introducing new metadata in the form of XMAP doucments concerning the mapping rules of the schema of the wrapped database.

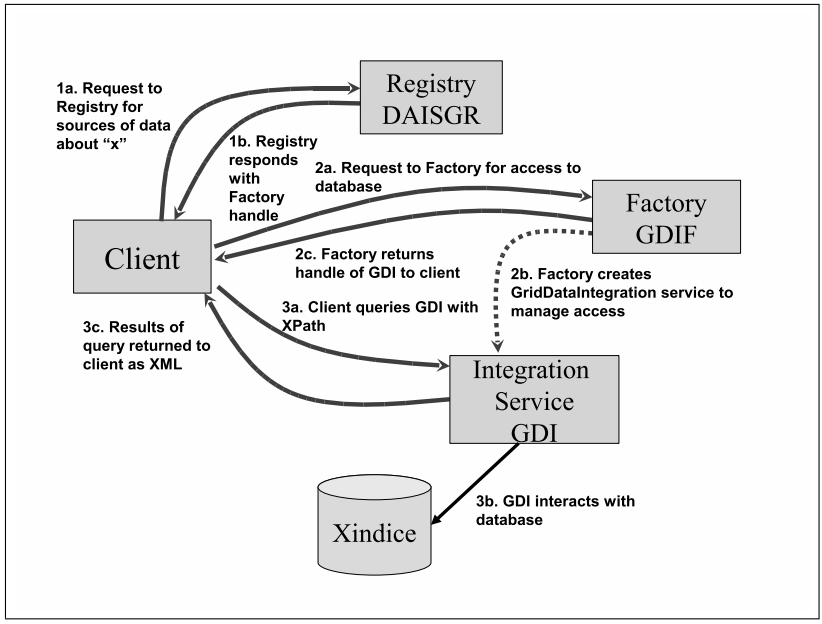


Figure 6. OGSA-XDI component services.

3. *Database Access and Integration Service Group Registry* (DAISGR) which allows clients to search for GDIFs and GDIs that meet their requirements. This registry service is the one provided by the OGSA-DAI system, as no modifications are necessary to manage the introduced integration services.

3.1. The XPathQueryReformulation *Activity*

Activities provide the core functionality in the OGSA-DAI framework. They also provide extensibility points where implementers may add functionality not provided by the base OGSA-DAI distribution and yet will still work within the OGSA-DAI framework.

In order to extend the functionality of OGSA-DAI and still be able to inherit the standard one, we re-implemented the `XPathStatement` activity handler. By this, we seamlessly integrated the reformulator engine component of the data integration framework, pictured in Figure 3, into the activity-chain processable by the OGSA-DAI enactment engine. The new implemented activity is called `XPathReformulation` activity and is able to handle XML code valid against the standard OGSA-DAI `XPathStatement` type. Both the `XPathReformulation` and the `XPathQuery` activities are configured as query activities, that is activities directed against a data resource allowing for extraction of a data set from that resource. More precisely, what differentiates the two activities is that the `XPathReformulation` activity before really access data sources, performs schema integration by exploiting the mappings of the XML schema over which the XPath query has been formulated. So, each time a user invokes in the perform document the `XPathStatement`, the `XPathStatement` activity handler activates the `XPathReformulation` activity.

Every activity has an XML-Schema (see Figure 7) that defines the inputs and the outputs of the activity as well as any additional configuration details. Occurrences of

activity elements are used within perform documents to indicate which activities should be performed by an OGSA-XDI service. Further, every activity has an implementation class that performs the action associated with the activity, in our case, reformulating an XPath query using the XMAP reformulation algorithm.

```
<?xml version="1.0" encoding="UTF-8"?> ...
 <xsd:schema ...>
  <xsd:complexType name="XPathReformulationType">
    <xsd:complexContent>
      <xsd:extension base="gds:ActivityType">
        <xsd:sequence> ...
          <xsd:element name="expression" type="xsd:string"/>
          <xsd:element name="resourceSetStream">
            <xsd:complexType mixed="true">
              <xsd:complexContent mixed="true">
                <xsd:extension base="gds:ActivityOutputType"/>
              </xsd:complexContent>
            </xsd:complexType>
          </xsd:element>
        ...
  </xsd:complexType>
  <!-- xPathReformulation element -->
  <xsd:element name="xPathReformulation"
               type="gds:XPathReformulationType"
               substitutionGroup="gds:activity"/>
</xsd:schema>
```

Figure 7. Activity schema.

After having implemented an activity it is necessary to configure it. Configuration is achieved using an activity configuration document. The `XPathReformulation` activity configuration document (see Figure 8) includes information on the activities that OGSA-XDI services configured with this document support.

```
<activityConfiguration>
  <activityMap>
    <activity name="xPathReformulation"
              implementation="gdis.XPathReformulationActivity"
              schema="xpath_reformulation.xsd">
      ...
      <property key="daisgr"
                value="http://localhost:8080/ogsa/services/
                              ogsadai/DAISGR"/>
    </activity>
  </activityMap>
</activityConfiguration>
```

Figure 8. Fragment of the Activity configuration document.

Our activity is instantiated when the inherited GDS enactment *engine* validates the submitted *perform* document. The engine loads the handler for this activity specified in

the configuration file for the GDI, e.g. via the `activityMap` element, and instructs the activity by passing on the appropriate element of the perform document to the activity, in our case the `XPathStatement` element containing the XPath query to reformulate.

3.2. GDIF: The GDI Factory Service

A Grid Data Service Factory (GDSF) is a specialised factory service which exposes a data resource as a service and can create new Grid Data Services (GDS) through which clients can interact with this data resource. Moreover, clients can query a GDSF to extract information about the data resource exposed by the GDSF and so determine whether the data resource meets their application-specific requirements. A GDSF allows other clients to access information about its state, including information on the data resources it exposes and the Grid Data Services it can therefore create. This access is provided via the `FindServiceData` operation.

The introduced GDIF factory service implements the same portType of a GDSF, the `GridDataServiceFactoryPortType` portType, extending it with new Service Data Elements (SDEs) concerning the mapping rules pertinent to the schema of the exposed XML data source.

GDIFs advertise information on the GDIs they can create using a DAISGR. A client can contact a GDIF and request creation of a GDI serving the particular data resource associated with the GDIF.

The capabilities of a GDSF and the GDSs it can create are specified by a configuration document, the *data resource configuration document* (see Figure 9).

```
<?xml version="1.0" encoding="UTF-8"?>
<dataResourceConfig
    ...>
  <metaData>
    <productInfo>
      <productName>Apache Xindice</productName>
      ...
    </productInfo>
    ...
    <schemaMapping>
    <XMAP>
     <sourceSchema>S2</sourceSchema>
     <Rule cardinality="Mapping1-1">
      <destinationSchema>S3</destinationSchema>
      <sourcePath>/S2/flight/route/destination</sourcePath>
      <destinationPath>/S3/trip/itinerary/location</destinationPath>
     </Rule>
      ...
    </XMAP>
    </schemaMapping>
  </metaData>
  ...
</dataResourceConfig>
```

Figure 9. Data resource configuration document.

3.3. GDI: The Grid Data Integration Service

The Grid Data Integration Service (GDI) is the central OGSA-XDI component. Differently from a GDS, a GDI other than supports data access and data delivery functions, it further provides XML data integration facilities. Similarly to a GDS, each GDI exposes a single XML data resource. Each GDI is created by a GDIF which manages access to that data resource.

Analogously to the OGSA-DAI Grid Data Service (GDS), the GDI supports a document-oriented interface for database requests. Therefore, all accesses and updates to the state of a database are carried out by way of a `perform` operation, the parameter of which is a request document: a client submits a data retrieval or update request in the form of an XML document, the GDI executes the request and returns an XML document holding the results of the request.

Every GDI instance introduces specific portTypes:

1. The `Import Mappings (IM)` portType, through which a client can import the mappings stored in the other GDI services in the system. The imported mappings will be exposed by the GDI as SDEs.
2. The `Query Reformulation Algorithm (QRA)` portType. The `QRA` portType extends the GDS `GridDataService (GDS)` portytpe, so it facilitates the document-oriented interaction between a client and a GDI. Analogously to the `GDS` portType, the `QRA` portType provides one operation, `perform`, which accepts a *perform* document (see Figure 10). This document describes the operations the client wishes the GDI to perform. In order to execute a data integration operation, we can invoke the `perform` operation of the `QRA` portType that through the `XPathQueryReformulation` activity invokes the XMAP reformulation algorithm. The `perform` operation receives as input a query, schema mappings and produces zero, one or more reformulated queries. The reformulated query will be the input of the `GDS` portType offered by the OGSA-DQP GDQS service.

```
<?xml version="1.0"?> <gridDataServicePerform>
 ...
  <xPathStatement name="statement">
    ...
    <expression>/S1/trip[year = "2000"]/route/location</expression>
    <resourceSetStream name="statementOutput"/>
  </xPathStatement>
</gridDataServicePerform>
```

Figure 10. Example of *perform* document.

3.4. Service Interactions

Typical interactions in the system are those that take place when the two following activities are performed: (i) joining of new resources in the system, (ii) submitting a query with

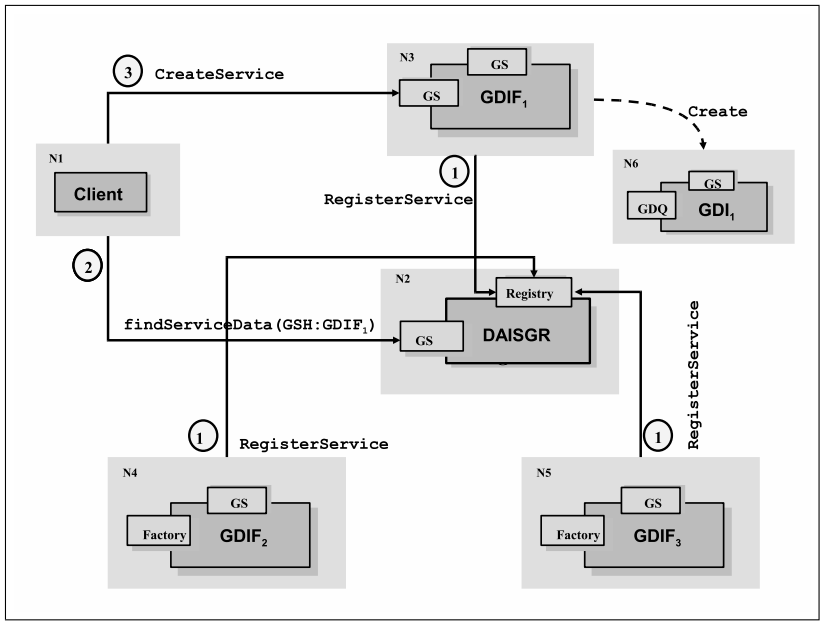


Figure 11. Factory Registration and creation of a GDI instance.

consequent execution. We distinguish between database administrator-side interactions and database client-side interactions.

The first type of interaction, the joining of new resources in the system, is an administrator-side interaction. When a database joins the system, the database administrator has to do the following actions:

1. Create the Grid Data Integration Factory (GDIF) and configure it to represent a single XML database accessed via the XIndice database management system (see Figure 11).
2. Create the data resource configuration document to configure the GDIF with the data resource that the GDIs it creates will have access to. The configuration document specifies the activities supported by these GDIs and the XML database metadata.
3. Create the XMAP mappings of the schema of the exposed data source stored as SDEs of the configuration document. To this aim the database administrator may need to look at both the schemas and the mappings of the other databases in the system. Nevertheless, he may want to share the mappings already in the system. Therefore, the database administrator may execute the following operations:

   (a) Import logical and physical schemas of the participating data sources through the `importSchema` operation of the `GDQ` portType (see Figure 12).
   (b) Import the mappings established in the system via the `importMappings` operation of the `IM` portType (see Figure 13).

As said before, the other typical interaction in the GDIS system is query execution. A typical query request scenario can be briefly outlined as follows.

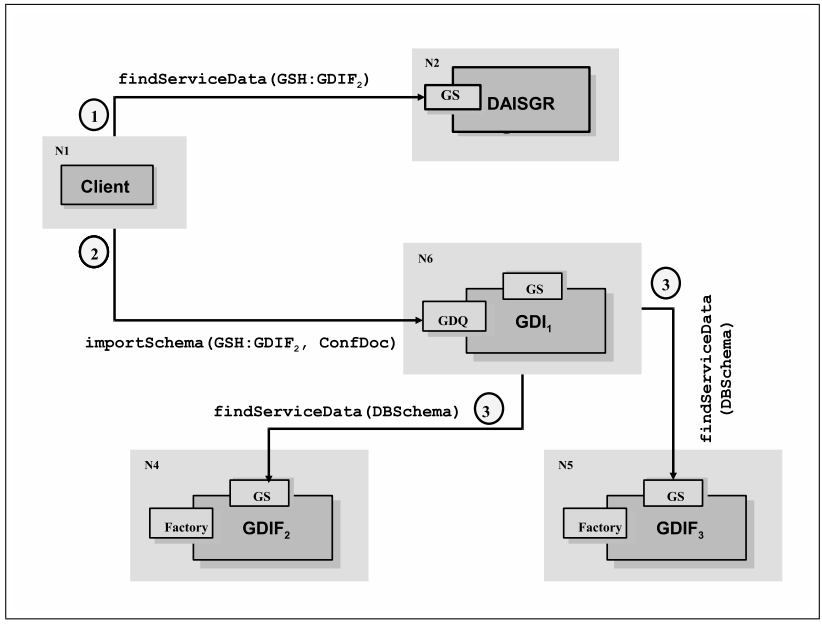


Figure 12. Import schemas.

1. Discover of the factory that can be used to create the desired actual service instance (see Figure 11). Through the `FindServiceData` operation the client asks the `GS` portType of the DAISGR that will return to it the address of a GDIF (the factory of the GDI Grid Service).
2. Create the GDI instance by using the `CreateService` operation of the `factory` portType implemented by the GDIF service.
3. Before writing a query a user may wish to access schemas of data sources. To this aim, the user can choose which database schemas to import through the `importSchema` operation of the `GDQ` portType of the GDI service (see Figure 14).
4. The client then can formulate the query and send it to a reformulator engine through the `QRA` portType of the GDI service. The `QRA` portType implements the `perform` operation that executes the XMAP reformulation algorithm receiving as input the query, in the form of an XML document, and importing the mappings established in the system via the `importMappings` operation provided by the `IM` portType of the GDI service.
5. Each produced reformulated queries is then transmitted, in the form of an XML document, to the GDQS service via the perform operation of the GDS portType. The next interactions are the same as those of a typical OGSA-DQP execution and hence the partitions of the distributed query execution plan are scheduled for execution at different GQES instances created by the GDQS. It should be noted that we allow for partial query answering, thus it is not necessary to collect the sub-query answers together in order to obtain the overall query answer. This way, each sub-query result is passed to the GDI service.

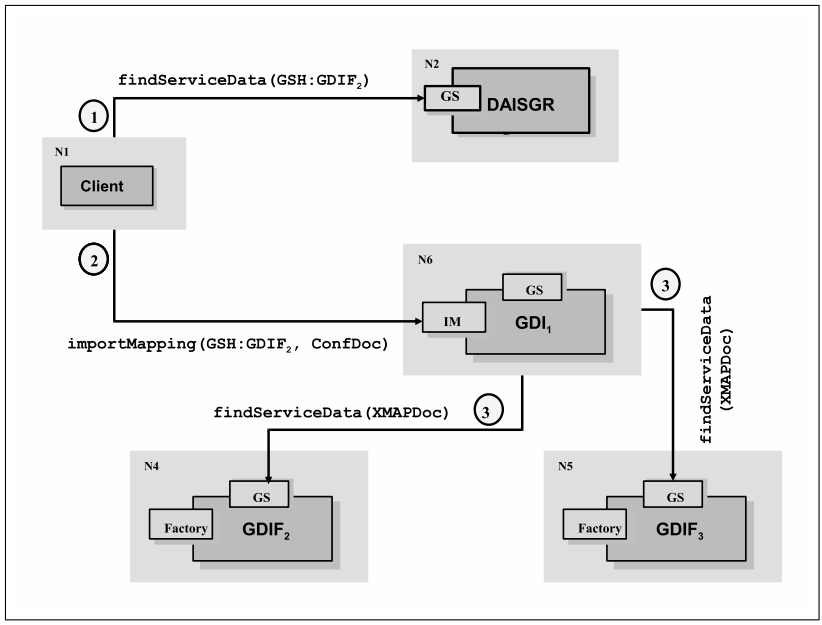


Figure 13. Import mappings.

4. GDIS Evaluation

In this section we briefly present an experimental evaluation of the GDIS prototype.

To experiment with GDIS, we have setup two GDIS systems, one located on LAN and the other one over the Internet. The characteristics of the machines and network involved in both settings are detailed in Figure 15.

In these experiments, we focus on the performance of GDIS as a whole, not on the specificity of the reformulation process which has been detailed in a previous work. The results presented in this section were produced according to the following protocol: the client uses the factory service (GDIF) to create a GDI instance. That instance is then used to submit the same request 100 times (sequentially), and finally destroys the instance it has created.

Measurements were performed using passive network capture analysis techniques. Packet sniffer were installed on each of the involved machines and HTTP requests and responses were reconstructed from the raw packet traces. Combining the traces collected at the different end points provides us with one the finest achievable accuracy, without needing to modify the software under scrutiny.

Reformulating queries that are submitted to it is the real task of our GDI. Interactions taking place in this process are represented in Figure 16. First, the client requests to the GDIF the creation of the GDI instance. After some time spent to prepare the perform document, the client submits it to the instance it has just created. During the reformulation process, the GDI asks the registry for any mapping information it might need. It is visible on the figure that mapping information queries are performed lazily, only when it is first needed. This is why we don't see a bunch of requests to the registry and then some quiet time to perform reformulation. Instead, reformulation CPU time is spread in

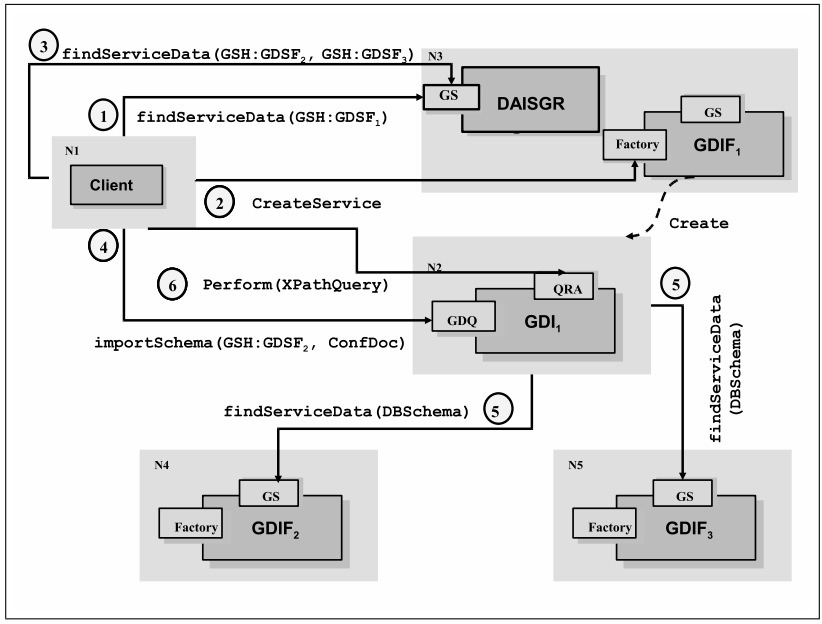


Figure 14. Query Request.

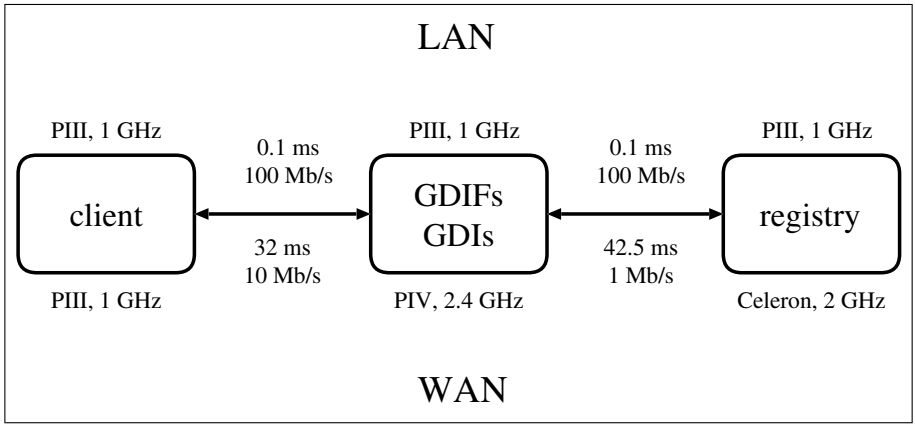


Figure 15. Characteristics of the 2 configurations (LAN and WAN) used to experiment with the GDIS system. For each link, the average round-trip times and bandwidth are given, and for each host, the CPU.

between mapping information requests: as soon as a new reformulated query is found, the GDI asks the registry for the XMAP document of the schema over which the query is expressed (assuming it has not seen it before). We shall also notice that XMAP documents are all stored on the registry and that the GDI therefore does not interact with the other GDIFs to obtain mapping information.

Understanding the dynamics of the query reformulation is of high importance to let us understand the results of the experimentations we have performed.

We evaluated the query reformulation time in both the LAN and WAN scenarios. As XMAP evaluation is out of the scope of these tests, we will focus on the cutoff of the

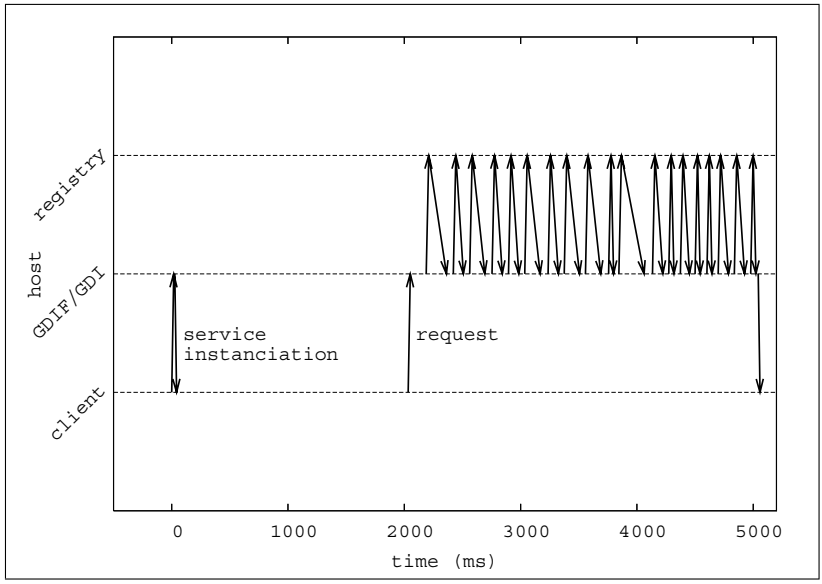


Figure 16. Entity interactions during the reformulation of a specific query in the WAN configuration. Interactions are also in this case HTTP requests and responses.

time spent in the system.

4.0.1. LAN Case.

Figure 17 shows the results (relative values) of the experiments when all entities involved in the system are located on the same LAN. Here has been used a mapping set with an average rank of 2. The GDI time here is really the time used by XMAP to perform the reformulation, while the "net" time is the communication time spent on the network asking and waiting for XMAP documents. The low-latency and high-bandwidth network connecting the different machines makes this time really low. At the maximum, it represents 6.7% of the total time spent for the reformulation. Network times are almost negligible compared to those required to perform query reformulation. The results obtained confirm the intuition that network time only depends on the number and the size of the XMAP documents required to perform the full reformulation.

4.0.2. WAN Case.

In the WAN configuration the dominant cost is represented by the network time. Results for a mapping set with average rank equals to 2 are shown in Figure 18. They make it obvious that even for a moderate round-trip time (42 ms on average) between the GDIF and the registry, network cost becomes largely predominant. From our results, it appears that the total network time is roughly equivalent to two round-trip times between the GDI and the registry, plus the time necessary to transmit the document at the speed of the wire, multiplied by the number of XMAP documents to import. With our settings, this was always lower than 3 round-trip times. This gives us useful hints to reduce the overall execution time of the reformulation.

Note that both the GDI and network times are dependent from the specific mappings. In fact, not only certain paths could involve a larger number of mappings, but

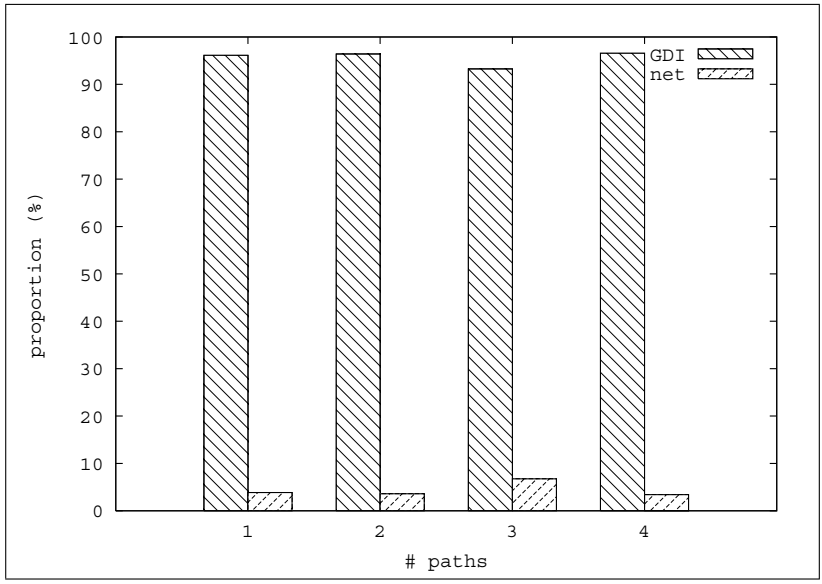


Figure 17. Relative repartition of the time spent for query reformulation with mappings of an average rank equals to 2 for a LAN configuration.

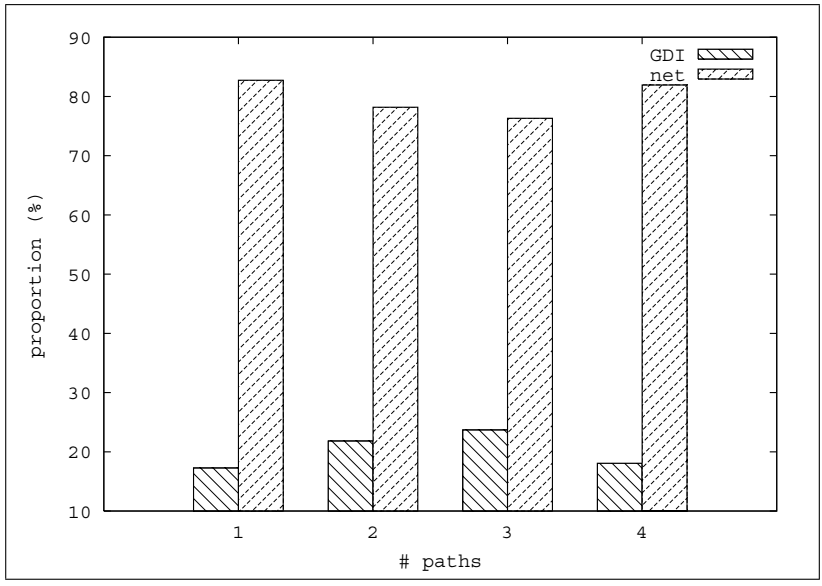


Figure 18. Relative repartition of the time spent for query reformulation with mappings of an average rank equals to 2 for a WAN configuration.

some mappings could provide better connectivity than others producing, thus, a larger number of reformulations of the original query. In the experiments shown in Figure 18, the number of mappings involved in the execution of the query composed of four paths is higher compared to the other queries and, consequently, the number and dimension of the

XMAP documents transmitted over the network increase. In such a case the percentage of network time compared to that of the GDI time is even more prevailing.

5. Conclusions

The continuously growing availability of data sources in Grid environments requires a coordinated and integrated access of such data due to the heterogeneity of the involved data models. Based on these reasons we proposed the XMAP framework and the GDIS architecture combining a data integration approach with existing Grid services for querying heterogeneous, distributed databases. This way we provide an enhanced, data integration-enabled service middleware supporting distributed query processing. As for future work, we plan to extend the XMAP framework in order to support more expressive query languages (e.g. XQuery). Further, declarative query approaches will be used to express the distributed and dynamic properties of the evolving network (it could change during query execution), following recent ideas on declarative networking.

Acknowledgements

This work has been partially supported by the FP6 Network of Excellence CoreGRID (Contract IST-2002-004265).

References

[1] A Gounaris, C Comito , R. Sakellariou and D. Talia A Service-Oriented System to Support Data Integration on Data Grids In: 7*th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, (2007) 627–635.

[2] Carmela Comito, Domenico Talia, Anastasios Gounaris, Rizos Sakellariou A Service-Oriented System to Support Data Integration on Data Grids Institute on Knowledge and Data Management, CoreGRID Technical Report, TR-0071, 26 Feb, 2007.

[3] The Globus Alliance (2004) The Globus toolkit, http://www.globus.org

[4] Foster I, Kesselman C, Tuecke S (2001) The anatomy of the Grid: Enabling scalable virtual organization. Int. Journal of Supercomputing Applications 15:200–222

[5] Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys **22** (1990) 183–236

[6] Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: VLDB. (1996) 251–262

[7] Calvanese, D., Damaggio, E., Giacomo, G.D., Lenzerini, M., Rosati, R.: Semantic data integration in P2P systems. In: DBISP2P. (2003) 77–90

[8] Halevy, A.Y., Suciu, D., Tatarinov, I., Ives, Z.G.: Schema mediation in peer data management systems. In: ICDE. (2003) 505–516

[9] Antonioletti, M., et al.: OGSA-DAI: Two years on. In: Global Grid Forum 10 — Data Area Workshop. (2004) http://www.ogsadai.org.uk/.

[10] Alpdemir, M.N., Mukherjee, A., Gounaris, A., Paton, N.W., Watson, P., Fernandes, A.A.A., Fitzgerald, D.J.: OGSA-DQP: A service for distributed querying on the grid. In: EDBT. (2004) 858–861

[11] Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R., Vetere, G.: Hyper: A framework for peer-to-peer data integration on grids. In: ICSNW. (2004) 144–157

[12] Brezany, P., Woehrer, A., Tjoa, A.M.: Novel mediator architectures for grid information systems. FGCS - Grid Computing: Theory, Methods and Applications. **21** (2005) 107–114

[13] Comito, C., Talia, D.: XML data integration in OGSA grids. In: Proceedings of the First VLDB Workshop on Data Management in Grids (DMG'O5). (2005) 4–15

[14] Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The physiology of the grid. Global Grid Forum (2002) http://www.globus.org/alliance/publications/papers/ogsa.pdf.
[15] Comito, C., Talia, D.: GDIS: A service-based architecture for data integration on grids. In: GADA. (2004) 88–98
[16] Foundation A (2002) Jakarta tomcat. http://jakarta.apache.org/tomcat/
[17] Czajkowski K, et al (2004) The WS-resource framework version 1.0. The Globus Alliance, Draft. http://www.globus.org/wsrf/specs/ws-wsrf.pdf
[18] Clark, J., DeRose, S.: XML path language (XPath) version 1.0. W3C Recommendation (1999) http://www.w3.org/TR/xpath.

Chapter 4

Grids and High Performance Computing in Action

This page intentionally left blank

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

The GRelC Project: state of the art and future directions

Sandro FIORE, Massimo CAFARO, Maria MIRTO,
Salvatore VADACCA, Alessandro NEGRO and Giovanni ALOISIO
CACT/NNL and University of Salento, Lecce, Italy
Euro-Mediterranean Centre for Climate Change, Italy

Abstract. Grids encourage and promote the publication, sharing and integration of scientific data, distributed across Virtual Organizations. Scientists and researchers work on huge, complex and growing datasets. The complexity of data management within a grid environment comes from the distribution, heterogeneity and number of data sources. Along with coarse grained services (basically grid storages, replica services, storage resource managers, etc.), there is a strong interest on fine grained ones concerning, for instance, grid-database access and management. Moreover, as grid computing, technologies and standards evolve, more mature environments (production grids such as EGEE) become available for production based activities and tools/services able to access in grid relational databases are also strongly required. We describe the Grid Relational Catalog (GRelC) Project, a framework for grid data management, highlighting its approach, architecture, components, services and technological issues.

Keywords. Grid Computing, GRelC Project, Grid Data Management, Data Access Service, Data Storage Service, Data Gather Service.

Introduction

Many scientific and engineering applications require access to large amounts of distributed data (terabytes or petabytes). Grid Computing [1] allows dynamically, securely and transparently aggregating a variety of computational and data resources (data storages and databases) geographically distributed over the Internet.

Current Production grids such as EGEE [2] or Teragrid [3] produce huge amounts of scientific data within a realistic short time using multiple computers deployed on the Internet, across different Virtual Organizations (VOs) [4].

Data Grids provide the proper foundations for Computational Grids, eliminating the data access bottleneck inherent in a grid topology and creating a unified view to widespread data sources (files and databases). In the last years research activities related to Grids have been generally focused on applications where data is stored on files; however, accessing and integrating legacy/new databases is becoming a fundamental issue. We describe the Grid Relational Catalog (GRelC) Project [5], discussing its history, main goals, grid data services and components, future work, etc.

The outline is as follows. In Section 1 we briefly recall some issues concerning data and computational grids, whereas in Section 2 we deal with the project's history. In Section 3 we describe the GRelC services, whereas in Section 4 we highlight GRelC

services interoperability with gLite-based grids. In Section 5 we present two projects using the GRelC middleware and finally we conclude this work in Section 6.

1. Grid computing: data and computational grids

The grid computing paradigm is widely regarded as a new field, distinguished from traditional distributed computing owing to its main focus on large-scale resource sharing and innovative high-performance applications.

The grid infrastructure can be seen as an ensemble of Virtual Organizations (VOs), reflecting dynamic collections of individuals, institutions and computational resources. It is basically made up of two components: computational and data grids.

A *computational grid* (Fig. 1) provides the computing power needed to execute highly demanding, CPU-bound applications, whereas a *data grid* provides data management services that enable data access, integration, synchronization, virtualization, distribution, etc. A data grid provides the foundations for a computational grid creating a unified view of disparate data sources. It supplies advanced capabilities and eliminates bottlenecks inherent in grid topologies.

Extreme performances in grid environments can be achieved if both the components are tightly implemented and integrated to support the application level. Thus, a data grid represents a fundamental building block and the GRelC project is strongly connected with this multifaceted and complex topic.

1.1. Grid Database Management Systems: Main approaches

Within the grid data management area, in the last two decades, several national and international research projects (EDG [6], SRB [7], etc.) addressed interesting topics regarding files (*coarse grained approach*). The output of these research activities was primarily connected with grid storage services [8], replica manager/consistency/catalogue systems [9], storage resource managers [10], enhanced grid file transfer protocols [11], etc.

After 2000 there was an increasing need and interest in grid database management systems (*fine grained approach* [12],[13],[14],[15]), i.e., middleware to grid-enable legacy and/or new databases running on top of well known DBMSs [16] (Oracle [17], IBM DB2 [18], MySQL [19], PostgreSQL [20], etc.).

Accessing databases on the grid became a very challenging issue and several research teams tried to address this problem. New scenarios (involving grid-enabled databases running within existing computational grids) that could not be previously performed owing to the lack of grid middleware, became feasible, including advanced data management activities (data mining, data access, data integration/federation, data warehousing, etc.).

In the last few years, several approaches for accessing databases on the Grid were developed. The two most important ones consist on (i) providing new grid services or (ii) making grid-aware existing DBMSs.

Within the former case fall research projects such as Spitfire [21], OGSA-DAI [22], [23], Eldas [24] and GRelC [25] with the main common goal to provide a transparent, secure and robust grid interface to existing DBMSs. The rationale behind is that no new DBMS need to be developed, only grid services that: (i) provide a grid interface to DBMSs, (ii) require no changes to existing databases, (iii) conceal the DBMS

heterogeneity and the physical database location, (iv) work seamlessly with other existing grid components, (v) support data transfer and security protocols (GridFTP and Grid Security Infrastructure [26]), (vi) provide uniform interfaces independently of the back-end DBMS, (vii) are loosely coupled to existing DBMSs, etc.

Obviously, these projects exhibit many inherent differences related to design, performance, security, supported platforms, provided features, interoperability and compliance with existing middleware (gLite [27], GLOBUS [28], etc.) and standards (OGSA, WSRF [29], etc.), current support, etc. that definitively make all of them rather different.

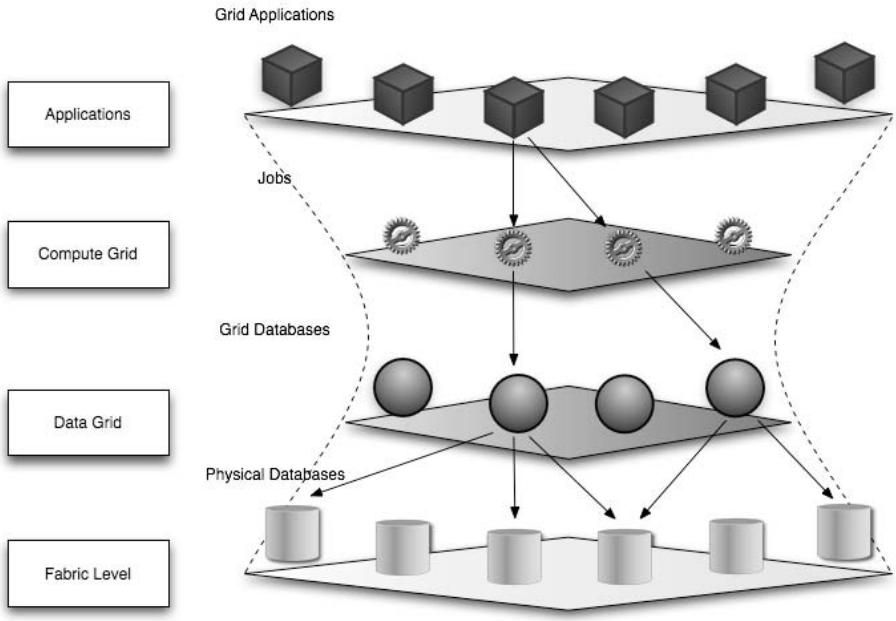


Figure 1: Hourglass model

In the latter case we mainly find some efforts connected with existing DBMSs such as Oracle 11g, IBMDB2 and MySQL, which try to enclose within their products grid support such as GSI (Grid Security Infrastructure), etc.

This approach has an inherent drawback, that is, it is tightly coupled with and does not leave out of consideration the underlying DBMS. Consequently, for each DBMS the same kind of support should be replicated, whereas in the grid service approach the same service interacts with a set/all of the DBMSs.

As stated before, the GRelC Project falls within the first category and tries to address several important requirements: first of all extreme performance, as well as a high level of transparency, security including full GSI/VOMS (Virtual Organization Membership Service) support, robustness and interoperability with regard to modern standard and existing middleware.

2. GRelC Project: a bit of history

The Grid Relational Catalog project started in 2001 (Fig. 2) to address specific issues concerning grid database management systems.

The first service developed within the project was the Dynamic Grid Catalog Server [30], i.e., the ancestor of the GRelC DAS. We considered as proof of concept a relational database containing the same information stored within the LDAP-based Globus MDS (Monitoring and Discovery System). DGC was a successful attempt of relational GIS and subsequent work in the same area [31], [32], [33], demonstrated that the relational approach for Grid Information Systems could be considered suitable and efficient. The super-peer DGC-IS was also presented in [34] demonstrating a high level of performance, scalability and manageability w.r.t. the MDS hierarchical approach.

Thereafter, the DGC laid out the foundations for the first prototype of GRelC Server [35], a Grid Database Access Service. The aim of this service was to access databases on the grid, (i) providing security, efficiency and transparency, (ii) trying to outperform in grid environments direct database connections, (iii) and providing users with advanced query support. This service, leveraging a client/server model, provided:

- a set of command-line clients for end-users;
- a wide client library for developers [36];
- a proprietary protocol for client/server communication;
- a GSI enabled GRelC Server, with GridFTP and compression mechanisms for remote data delivery.

The first prototype was deployed on the European SPACI grid sites and initial performance tests were carried out on the GridLab project testbed [37].

The second service (GRelC Gather Service [38]) provided by the GRelC Toolkit was developed to supply data federation capabilities. It represented a second layer of virtualization and it was successfully used to support a set of applications concerning biomedicine (virtual distributed clinical folder [39]) and earth observation systems (grid enabled web map server [40]). It is worth noting here that both the GRelC Server and GRelC Gather Server leveraged a client/server model, with a proprietary communication protocol between client and server; interoperability with other external services/components was not straightforward.

In the same period, there was a migration towards OGSA (Open Grid Service Architecture) and Web Services, therefore (as the majority of the research projects) the GRelC Project adopted the new service oriented paradigm to strongly address data virtualization and service interoperability. In this second phase (WS oriented) there was a reengineering of the two services that were renamed: GRelC Data Access Service (DAS) and GRelC Data Gather Service (DGS). Interoperability was eventually addressed successfully.

The GRelC DAS introduced new functionalities with regard to the past connected with Grid-DataBase management (dump of a database, database relocation, virtual space concept/management, etc.), security (support for both GSI and VOMS), user management, etc.

An inherent drawback of the first GRelC DGS release was its centralized approach, which did not provide a high level of scalability as required by a grid environment. On the contrary, the second release (that we will detail in the next section) improved availability, scalability and transparency leveraging a peer to peer approach and exploiting a high level of virtualization.

Finally, we developed a first prototype of GSI-enabled storage service (GRelC Data Storage Service, DSS) to provide a lightweight disk storage management solution. It is currently used within the SEPAC [41] research grid for bioinformatics "in silico" experiments [43].

GRelC DAS, DGS and DSS currently belong to the GRelC production release which is also gLite-compliant [44] to enable tight interaction and interoperability with the main services currently deployed on EGEE, the most important gLite-based European production grid.

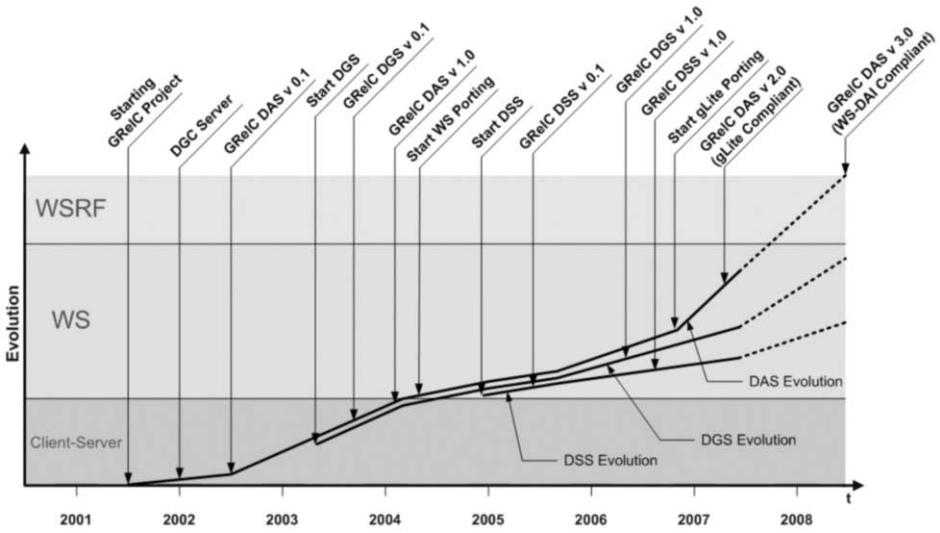


Figure 2: GRelC Project evolution

Research activities are currently related to the development of a WS-DAI, WS-DAIR, WS-DAIX [45] compliant release of the GRelC DAS in order to address/adhere to new OGF standards for grid database access and management. This is an ongoing work and a related preliminary release is in alpha test.

3. GRelC Services: DAS, DSS, DGS

In this Section we briefly provide some details about the three main grid services released within the GRelC Toolkit:

- GRelC Data Access Service (DAS), to access and interact with relational and non-relational databases;
- GRelC Data Storage Service (DSS), to manage workspace areas, reliable file transfer, POSIX compliant operations, etc.;
- GRelC Data Gather Service (DGS), to federate widespread data sources leveraging a P2P approach.

The three services are entirely developed utilizing the C programming language as GSI/VOMS enabled web services to address efficiency, security and robustness; they share some design/technological issues that will be described in the following subsection. The three proposed services provide a data grid framework to (i) manage

data sources (databases, *fine grained* approach) and storages (files, *coarse grained* approach) and (ii) to integrate metadata information by exploiting a P2P DGS-based overlay network. This data grid infrastructure has been successfully used for EOS and bioinformatics experiments.

3.1. Common foundations

The three services are similar w.r.t.:

- technological issues: from a technological point of view the three services are web services developed exploiting the gSOAP Toolkit [46]. All of the provided methods are listed within service-related WSDL documents;
- security: to establish a secure data communication channel, we adopted the Globus Toolkit GSI, (integrated as a gSOAP plug-in) [47]. Moreover, grid roles and VO membership management in grid environments are provided by means of the VOMS support [48];
- performances: to address efficiency, the three services are WS-I based multi-threaded web services, developed in C. As pointed out by several work (mainly addressing GRelC DAS performances), this requirement has been successfully achieved and it represents a key feature of the proposed toolkit/services;
- user management: it is based on a two level approach, basically leveraging three kinds of users: root, administrator and end-user. The first one manages a service and she can designate new administrators. Administrators in turn can configure a service; there can be different levels/roles depending on the underlying service (DAS, DGS, or DSS). Finally, end-users are the ones that access the services to perform queries (DAS), file transfers (DSS) or data federation activities (DGS). Also in this case there can be different kinds of end-users depending on the existing roles and data access control policies;
- metadata catalog: we designed and implemented a relational metadata catalog to manage service metadata information. All of the key metadata (users, groups, VOs, policies, etc.) are stored server-side and can be selected/joined/filtered depending on the underlying operations. For efficiency reasons we will exploit an embedded relational DBMS both to further enhance performances and to remove the dependence on an external server.

In the following subsections we provide a brief description of the three services.

GRelC DAS

The GRelC DAS (Fig. 3) acts as a standard front-end for database access on the grid (we define a Grid-Database as a database instance managed by the GRelC DAS). This service provides both basic and advanced primitives to transparently access, query, manage and interact with different data sources, concealing the back-end heterogeneity, GSI/VOMS security details, SOAP connection and other low level issues.

Currently, the GRelC DAS provides (by means of wrappers developed by using proprietary libraries integrated within the GRelC SDAI, Standard Database Access Interface) dynamic binding to several physical RDBMSs such as Oracle, PostgreSQL, IBM DB2, MySQL, SQLite, Microsoft SQL Server, Sybase, UnixODBC. The security of the connection between DBMSs and GRelC DAS relies on Transport Layer Security

(TLS). The UnixODBC wrapper also allows the GRelC DAS to access textual databases or other UnixODBC-supported DBMSs.

To date, the GRelC DAS supports the following operations:

- data access functionalities related to grid database (i) bind, (ii) query and (iii) unbind. Concerning query management additional details can be found in [49]. It is worth noting here that, as pointed out by experimental results in [49], new kinds of queries designed for this service (leveraging compression mechanisms and chunking) can outperform direct database connections;
- user management functionalities to allow (i) adding and (ii) deleting users to/from a grid database, (iii) granting and revoking privileges (both for root, administrators and end users), (iv) managing user's profiles, etc.;

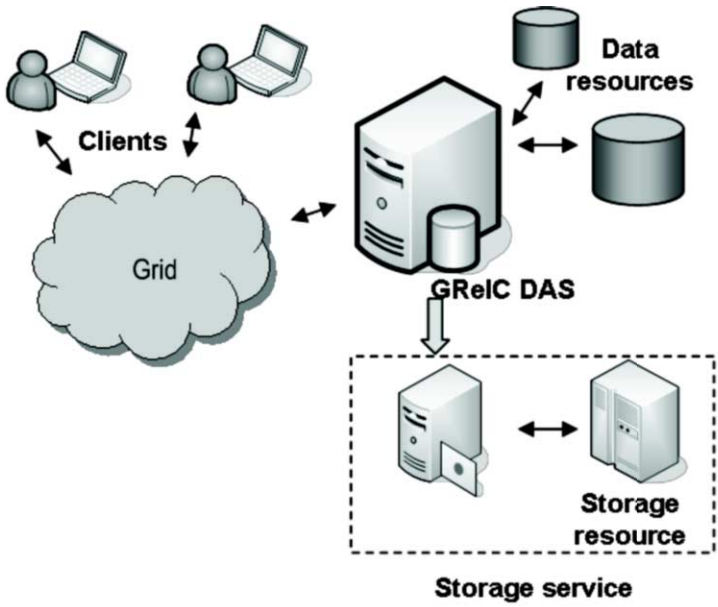


Figure 3: GRelC Data Access architecture in the large

- grid database functionalities related to (i) Grid-DB creation, (ii) registration, (iii) drop, and (iv) management. By registration we mean the possibility to map an existing database onto a grid-DB, by automatically importing all of the database related metadata (tables, attributes, types, domains, etc.).

GRelC DSS

The GRelC DSS (Fig. 4) is a novel data grid storage service that efficiently, securely and transparently manages collections of data (files) on the grid promoting flexible, secure and coordinated storage sharing, across Virtual Organizations (VOs).

A key concept is the *workspace*, a virtualized and grid-enabled space in which a community of users (for instance a VO or a cross-VO group) can share their files/folders. It represents a way to define a storage space accessible by principals (authorized grid users) sharing some common interests (i.e. a bioinformatics

experiment, a VO, etc.). Within a GRelC DSS, data is organized into workspaces; for each one we must define, among the others, the root user, a set of authorized users, groups and VOs, the workspace administrators, etc.

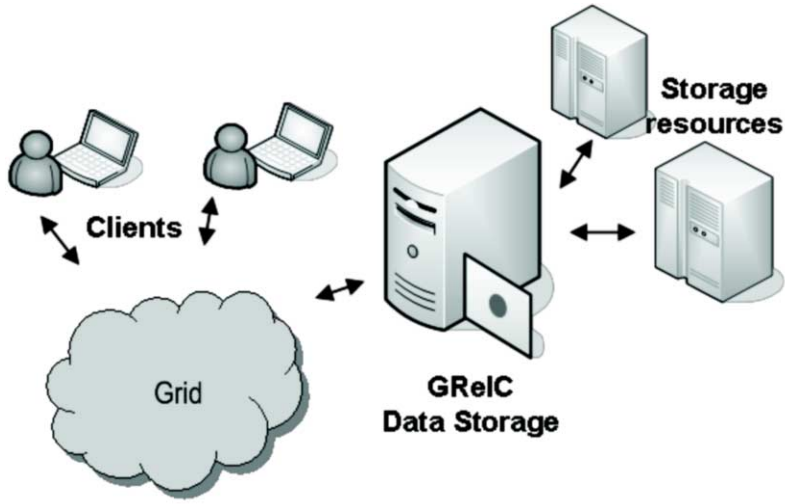


Figure 4: GRelC Data Storage architecture in the large

Within a workspace users can:

- transfer files and folder (common upload and download operations);
- perform Posix-compliant operations such as rename, delete, open, etc. on both files and folders;
- define the scope (visibility) associated to selected objects (e.g. files, folders, etc.);
- manage user's data access policies at a very fine grained level;
- perform reliable file transfer (RFT) operations (RFT is an asynchronous service primarily intended to be used to reliably transfer files from a storage resource to another one). It is worth noting here that the RFT service is not the one provided by the Globus Toolkit v4; it has been developed from scratch long before the Globus service was released and represents an important building block within a data grid environment and a key element of the GRelC DSS, because it (i) allows *reliably* performing file copy, (ii) supports scheduled *backup* operations among groups of storage resources and (iii) finally it provides an asynchronous service for generic (transactional or not) data management operations (copy, copy-parallel/partial, delete, rename, etc.).

GRelC DGS

The GRelC DGS (Fig. 5) represents the key component of the P2P GRelC Data Gather architecture and it acts as a peer – intermediate node – within the GRelC data management P2P architecture. A crucial point is that it exploits the data source virtualization concept de-coupling routing issues from the ones related to data resource access. This way the same overlay P2P grid architecture can be used within several

scenarios involving different data resources (files, databases, storage, etc.). This aim is achieved by defining the virtual resource concept (a set of well known APIs associated to several resource-specific implementations).

Moreover, the GRelC DGS implements the *project* concept, i.e., a collection of data organized into a fixed schema with a specific set of privileges and accessed by a group of users sharing common interests. Some examples could be a digital library, a federated database, a distributed information service, a federated storage service, etc.

The GRelC DGS implements several kinds of queries (*local* and *grid*) and two techniques (*routed response* and *routed response with gather processing*) to return a matching query result to a DGS node; moreover, management of Time-To-Live (TTL) and Hops-To-Live (HTL) parameters implemented within the system allow decreasing the network load.

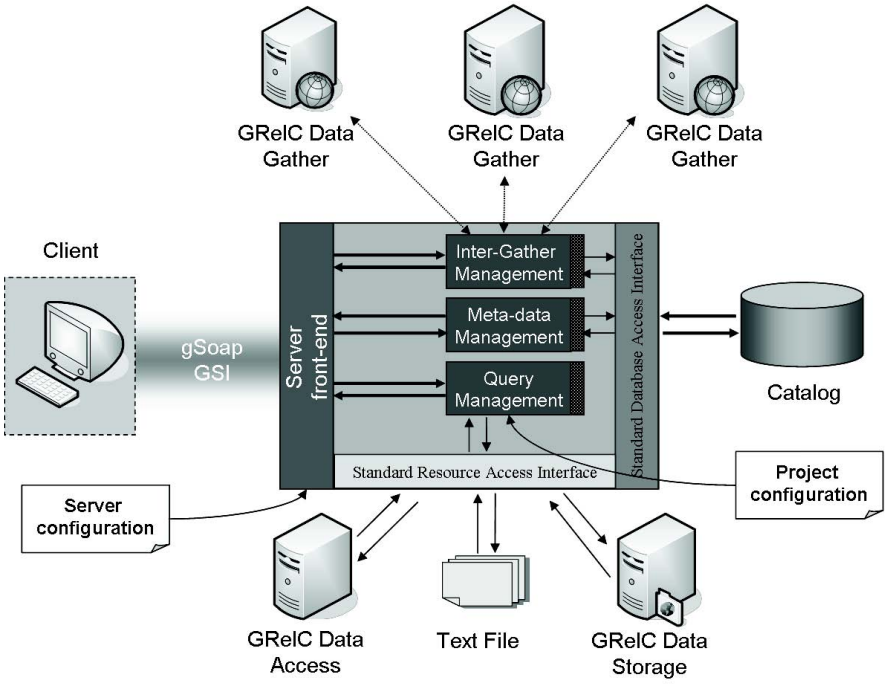


Figure 5: GRelC Data Gather architecture

Additional details regarding this service, query management, security and technological issues can be found in [38].

3.2. User Support: XGRelC and WebGRelC

The GRelC Services include a complete set of command-line clients for query submission (DAS), file transfer (DSS), distributed query (DGS), etc. Moreover, for each service we provide an X11 Qt based graphical interface (named XGRelC), which simplifies the interaction with the proposed services.

Recently, a web based environment has been also developed to provide seamless and ubiquitous data access, integration and management. The first prototype along this direction is WebGRelC [50].

The grid portal architecture follows a standard three-tier model and it will soon integrate user management functionalities as well. The three tiers within our architecture are the following ones: a *client browser* (first tier) that can securely communicate with a web server over an HTTPS connection (no other specific requirements are requested); the *Web Server* (second tier) which implements the *WebGRelC Portal* leveraging (i) GridSphere portlet container and (ii) GRB [51], GRelC DSS, DAS and DGS client libraries. It interacts with MyProxy [52] servers for secure user's credentials (proxy) storage and retrieval and with the Portal Metadata Catalogue to manage user's profiles. Finally, the *GRelC Services* are deployed on the third tier, the data grid infrastructure.

4. Interoperability with gLite-based grids

Currently, the EGEE middleware (gLite) lacks ad hoc software for grid-database access. The latest GRelC DAS release tries to bridge the existing gap, providing a well suited solution, strongly integrated with the available gLite components (Computing Element, Worker Nodes, Resource Broker, Storage Element, etc.).

The security/authorization framework we adopted (GSI/VOMS, both user and VO centric) is fully compliant with the EGEE one. Moreover, the GRelC DAS is able to publish (within the BDII [53], by means of a rich schema extension) a set of information connected with: grid databases, tables, fields, GRelC DAS contact string, authorized VOs, etc. This way gLite broker components can discover for their matchmaking phase the grid-databases available on the grid.

The GRelC DAS release (gLite compliant) is currently deployed on the Grid INFN virtual Laboratory for Dissemination Activities (GILDA) t-infrastructure [54]. Launched in 2004 by the EGEE partner INFN [55], GILDA is a fully operational Grid testbed devoted to dissemination activities; it provides both users and system administrators with a practical hands-on approach to gLite based Grid systems.

Within the EGEE context, GILDA acts as a crucial component of the project's t-infrastructure (training infrastructure) programme, helping to pass on knowledge and experience, as well as computing resources, to the scientific community and Industry.

A training version of this software is already available for tutorial activities on the Grid CT wiki [56]. The first deployment on GILDA concerns a small set of sites, but we already plan to extend this deployment (both for server and client side) on the other GILDA sites identifying interesting test case grid databases related to the bioinformatics domain.

5. Projects

In the following subsections we will describe the research and production grids that actually host an installation of the GRelC Toolkit, defining their focus, the involved partners and how the GRelC middleware impacts on their activities.

5.1. SEPAC

The SEPAC (Southern European Partnership for Advanced Computing) project [57] started in 2004 when the CSCS (Swiss National Supercomputing Centre), the University of Zurich and the SPACI (Southern Partnership for Advanced Computational Infrastructures) joined the ETH (Eidgenoessiche Technische Hochschule) of Zurich, the Hewlett-Packard and the CILEA (Consorzio Interuniversitario Lombardo per l'Elaborazione Automatica), who were the members of the international TAPAC (Trans-Alpine Partnership for Advanced Computing) project. The main goal of SEPAC is to build an efficient and reliable computing infrastructure from heterogeneous production-level components (machines, operating systems, applications) and use it as a computing solution for general scientific applications.

The focus is on general applications in academia and in industry sector. The design includes the new features of grid technology: middleware updating, secure file transfer, virtual organization management, grid data management, web services and portals. The first SEPAC infrastructure was based on Globus Toolkit 2.0.4. It served to gain the initial experience in gridifying simple applications and to make the first attempts toward the construction of a user-friendly web interface. Currently, the SEPAC Grid uses the GT 4.x pre-Web-Service version and its functionalities have been extended by using a new Information Service (iGrid, which internally leverages GRelC libraries) [2], the Grid Resource Broker (GRB) and the GRelC Data Management toolkit [3]. Several GRelC DAS installations allow grid users (i) directly submitting queries to biological databases as well as to medical ones by using the GRelC DAS command-line clients, or (ii) carrying out through the GRB portal complex applications (including parameter sweep and workflow simulations) that need access to the available scientific grid databases.

Today, the SEPAC Grid has become a Grid infrastructure for bioinformatics and medical applications, offering to its users a huge variety of applications such as, for instance, the BLAST suite, GAMESS, Gaussian, AutoDock, GROMACS and VASP. This success was achieved with the help and expertise of its partners, the Computational Chemistry Group at the University of Zurich, the CILEA, as a partner of two international projects LITBIO [58] and BioInfoGrid [59], the CSCS - coordinator of the SwissBioGrid [60], [61] project, and SPACI with the involvement in the LIBI [62] project.

5.2. LIBI

The Italian FIRB LIBI ("International Laboratory for Bioinformatics") project, funded by the MIUR, proposes the setting up of an advanced Bioinformatics and Computational Biology Laboratory focusing on the central activities of basic and applied research in modern Biology and Biotechnologies.

LIBI, has its headquarter in Bari (Italy), but it is conceived as a laboratory "without walls". LIBI aims at the creation of a virtual working space for the academic and industrial partners belonging to public and private institutions. All of the partners provide their infrastructures and utilize the platform tools and equipments to benefit from the capacity and expertise made available by the other partners. Therefore the LIBI proposes itself as an organizational unit able to gather and enhance multidisciplinary expertise to develop a technological platform fitting to the promotion

of research activity in Bioinformatics in Italy and granting the recognition and use at international level.

Two types of actors have equal responsibility in the LIBI: technological and bioinformatics partners. The technological research units (RUs) are: CINECA in Bologna, INFN Padova and Bari sections, SPACI/CACT-NNL,University of Salento, Lecce and IBM Semea Sud, an industrial partner. The scientific RUs are CNRBA (Istituto Tecnologie Biomediche, CNR, Section of Bari), UNIBO (University of Bologna), UNIMI (University of Milan) and CBMTS (Centro di Biomedicina Molecolare, Trieste).

LIBI mainly focuses on the following data management research activities:

- the construction and maintenance of genomic, proteomic and transcriptomic databases;
- the design and implementation of new algorithms and software for the analysis of genomes and their expression products;
- the development of new databases of pathogens relevant for man, animals and plants.
- the design, development and maintenance of a cell cycle database.

A set of GRelC DAS are currently deployed in Bologna, Bari, Lecce, Milan and Trieste to provide grid access to the aforementioned genomic and proteomic databases within the LIBI distributed laboratory. Moreover, a specific driver for IBM DB2 is also under testing to provide support and access to a federated data bank composed of about twelve bioinformatics databases (OMIM, EMBL, UniProt, UTRef).

6. Conclusions and future work

Nowadays, many grid applications need to access, share, and integrate huge amounts of data, distributed across heterogeneous and widespread grid resources. Data grids could allow applications to improve their performances and quality of results providing efficient, robust and secure data services for a computational grid environment.

In this paper we presented the state of the art of the Grid Relational Catalog project, which provides a set of high level and grid-enabled data services for relational and non relational repositories as well as files on storage devices. We reported on the Data Access, Storage and Gather Services progress, presenting main issues and common foundations. In future releases, we will adopt a Grid Services architecture paying special attention to the Open Grid Service Architecture and to the emerging WS-DAI, WS-DAIR and WS-DAIX OGF specifications.

References

[1] I. Foster, C. Kesselman, *The Grid: blueprint for a new computing infrastructure*, Morgan Kaufmann Publishers - 1998

[2] Enabling Grids for E-sciencE (EGEE) Project – [http://www.eu-egee.org/]

[3] The TeraGrid Project – [http://www.teragrid.org]

[4] I. Foster, C. Kesselman, J. Nick, S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed System Integration*. - [http://www.globus.org /research/papers/ogsa.pdf]

[5] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, *The Grid Relational Catalog Project*, Advances in Parallel Computing, *Grid Computing: The New Frontiers of High Performance Computing*, L. Grandinetti (Ed), pp. 129-155, Elsevier – 2005.

[6] The DataGrid Project – [http://eu-datagrid.web.cern.ch/eu-datagrid/]
[7] Storage Resource Broker – [http://www.sdsc.edu/srb/index.php/Main_Page]
[8] C. Baru, R. Moore, A. Rajasekar, M. Wan, *The SDSC Storage Resource Broker*, Proc. CASCON'98 Conference , Nov.30 - Dec.3, 1998, Toronto, Canada.
[9] RLS: Replica Location Service – [http://www.globus.org/rls/]
[10] Grid Storage Management WG (GSM-WG) – [https://forge.gridforum.org/sf/projects/gsm-wg]
[11] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Tuecke, *GridFTP* – Internet draft – [http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf]
[12] Database Access and Integration Services WG – [https://forge.gridforum.org/projects/dais-wg]
[13] N. W. Paton, M. P. Atkinson, V. Dialani, D. Pearson, T. Storey, P. Watson, *Database Access and Integration Service on the Grid*, Global Grid Forum OGSA-DAIS WG. Technical Report (2002)
[14] P. Watson, *Databases and the Grid*. Technical Report CS-TR-755, University of Newcastle, 2001
[15] V. Raman, I. Narang, C. Crone, L. Haas, S. Malaika, T. Mukai, D. Wolfson, C. Baru. *Data Access and Management Services on Grid*, Technical Report Global Grid Forum 5 (2002).
[16] E.K. Clemons, *Principles of Database Design*, Vol. 1, Prentice Hall, 1985.
[17] Oracle – [http://www.oracle.com]
[18] IBM DB2 – [http://www-306.ibm.com/software/data/db2/]
[19] MySQL – [http://www.mysql.com]
[20] PostgreSQL – [http://www.postgresql.org]
[21] The Spitfire Project - [http://edg-wp2.web.cern.ch/edgwp2/spitfire/]
[22] M. Antonioletti, M.P. Atkinson, R. Baxter, A. Borley, N.P. Chue Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N.W. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead. *The Design and Implementation of Grid Database Services in OGSA-DAI.* Concurrency and Computation: Practice and Experience, Vol. 17, Issue 2-4, pp. 357-376, 2005
[23] K. Karasavvas, M. Antonioletti, M.P. Atkinson, N.P. Chue Hong, T. Sugden, A.C. Hume, M. Jackson, A. Krause, C. Palansuriya. *Introduction to OGSA-DAI Services.* LNCS, Vol. 3458, pp. 1-12, May 2005.
[24] Enterprise Level Data Access Services (Eldas) – [http://www.edikt.org/eldas/]
[25] Grid Relational Catalog Project – GrelC – [http://grelc.unile.it]
[26] S. Tuecke, *Grid Security Infrastructure (GSI) Roadmap* - Internet Draft - [http://www.gridforum.org/security/ggf1_-200103/drafts/draft-ggf -gsi-roadmap-02.pdf].
[27] gLite: Lightweight Middleware for Grid Computing - [http://glite .web.cern.ch/glite/]
[28] The Globus Toolkit – [http://www.globus.org]
[29] The Web Service Resource Framework - [http://www.globus.org /wsrf]
[30] G. Aloisio, E. Blasi, M. Cafaro, I. Epicoco, S. Fiore, M. Mirto, *Dynamic Grid Catalog Information Service* - Proceedings of the First European Across Grids Conference, February 13-14, 2003 Santiago de Compostela (Spain), Lecture Notes in Computer Science, Springer-Verlag, N. 2970, pp. 198-205
[31] B. Coghlan, A.W. Cooke, A. Datta, A. Djaoui, L. Field, S. Fisher, J. Magowan, W. Nutt, M. Oevers, M. Soni, N. Podhorszki, J. Ryan, A.J. Wilson and X. Zhu, *R-GMA: A Grid Information and Monitoring System* - WP3, UK e-Science all hands conference, Sheffield, 2-4 September 2002.
[32] G. Aloisio, M. Cafaro, I. Epicoco, S. Fiore, D. Lezzi, M. Mirto, S. Mocavero, *iGrid, a Novel Grid Information Service* - Proceedings of the First European Grid Conference (EGC) 2005, LNCS 3470, Lecture Notes in Computer Science, Springer-Verlag, pp. 506-515, P.M.A. Sloot et al. (Eds.), 2005
[33] G. Aloisio, M. Cafaro, I. Epicoco, S. Fiore, D. Lezzi, M. Mirto, S. Mocavero, *Resource and Service Discovery in the iGrid Information Service* - Lecture Notes in Computing Science, ICCSA 2005: International Conference - Grid Computing and Peer-to-Peer Systems, O. Gervasi et al. (Eds.), Volume 3482, pp. 1-9, ISBN 3-540-25862-0, Singapore, May 9-12, 2005
[34] G. Aloisio, M. Cafaro, I. Epicoco, S. Fiore, M. Mirto, S. Mocavero, *A performance Comparison between GRIS and LDGC Information Services* - Proceedings of SCI2003, 27-30 July, Orlando, Florida, Volume XII, pp. 416-420, 2003 Best Paper in the session: Management and Information Systems
[35] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, *The GRelC Project: Towards GRID-DBMS* - Proceedings of Parallel and Distributed Computing and Networks (PDCN) - IASTED, pp. 1-6, February 17 to 19, 2004, Innsbruck, Austria
[36] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, *The GRelC Library: A Basic Pillar in the Grid Relational Catalog Architecture* - Proceedings of Information Technology Coding and Computing (ITCC), April 5 to 7, 2004, Las Vegas, Nevada, Volume I, pp.372-376
[37] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, *Advanced Delivery Mechanisms in the GRelC Project*, ACM Proceeding of 2nd International Workshop on Middleware for Grid Computing (MGC 2004), October 18 2004, Toronto, Canada, pp. 69-74.
[38] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, S. Vadacca, *GRelC Data Gather Service: a Step Towards P2P Production Grids*, Proceedings of 22nd ACM SAC 2007, Seoul, Korea, Vol. 1, pp. 561-565

[39] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, *A Gather Service in a Health Grid Environment,* CD-Rom of Medicon and Health Telematics 2004, IFMBE Proceedings, Volume 6, July 31 - August 05, Island of Ischia, Naples, Italy
[40] G. Aloisio, M. Cafaro, D. Conte, I. Epicoco, S. Fiore, G.P. Marra, *A grid enabled Map Server*, Proceedings of Information Technology Coding and Computing (ITCC 2005), IEEE Press, Volume I, pp. 298-303
[41] Southern European Partnership for Advanced Computing (SEPAC) – [http://www.sepac-grid.org]
[42] Southern Parternship for Advanced Computational Infrastructure (SPACI) – [http://www.spaci.it]
[43] S. Fiore, M. Mirto, M. Cafaro, G. Aloisio, *GRelC Data Storage: Lightweight Disk Storage Management solution for bioinformatics "in silico" experiments,* to appear in the Proceedings of the 20th IEEE International Symposium on Computer-Based Medical Systems (IEEE CBMS 2007), June 20-22, 2007, Maribor (Slovenia)
[44] S. Fiore, M. Cafaro, A. Negro, S. Vadacca, G. Aloisio, R. Barbera, E. Giorgio, *GRelC DAS: a Grid-DB Access Service for gLite Based Production Grids* – Submitted to Fourth International Workshop on Emerging Technologies for Next-generation GRID (ETNGRID 2007)
[45] M. Antonioletti, A. Krause, N. W. Paton, A. Eisenberg, S. Laws, S. Malaika, J. Melton and D. Pearson. *The WS-DAI Family of Specifications for Web Service Data Access and Integration.* ACM SIGMOD Record, Vol 35, No 1, pp. 48-55, 2006.
[46] R. Van Engelen, K. Gallivan: *The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks*. CCGRID 2002: 128-135
[47] G. Aloisio, M. Cafaro, D. Lezzi, R. Van Engelen, *Secure Web Services with Globus GSI and gSOAP*, Euro-Par 2003, 26-29 Aug 2003, Klagenfurt, Austria, LNCS, Springer-Verlag, N. 2790, 421-426, 2003.
[48] R. Alfieri, R. Cecchini, V. Ciaschini, L. Dell'Agnello, A. Frohner, A. Gianoli, K. Lorentey, F. Spataro, *VOMS, an Authorization System for Virtual Organizations*. European Across Grids Conference 2003: 33-40
[49] S. Fiore, A. Negro, S. Vadacca, M. Cafaro, M. Mirto, G. Aloisio, Advanced Grid DataBase Management with the GRelC Data Access Service, submitted to the Fifth International Symposium on Parallel and Distributed Processing and Applications (ISPA07)
[50] G. Aloisio, M. Cafaro, S. Fiore, M. Mirto, *WebGRelC: Towards Ubiquitous Grid Data Management Services*, Proceedings of the 2nd International Workshop on Grid Computing Environments, held in conjunction with SuperComputing 2006, November 12-13 2006, Tampa, Florida, USA
[51] G. Aloisio, M. Cafaro, G. Carteni, I. Epicoco, S. Fiore, D. Lezzi, M. Mirto , S. Mocavero, *The Grid Resource Broker Portal*, to appear in Concurrency and Computation: Practice and Experience, Special Issue on Grid Computing Environments
[52] J. Novotny, S. Tuecke, V. Welch. *An Online Credential Repository for the Grid: MyProxy* Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, Aug 2001, pp. 104-111.
[53] Berkeley Database Information Index (BDII) – [https://twiki. cern.ch/twiki/bin/view/EGEE/BDII]
[54] G. Andronico, V. Ardizzone, R. Barbera, R. Catania, A. Carrieri, A. Falzone, E. Giorgio, G. La Rocca, S. Monforte, M. Pappalardo, G. Passaro, G. Platania, *GILDA: The Grid INFN Virtual Laboratory for Dissemination Activities*, TRIDENTCOM 2005: 304-305
[55] INFN Grid - [http://grid.infn.it/]
[56] GRelC Data Access Service tutorial on GILDA t-Infrastructure [https://grid.ct.infn.it/twiki/bin/view/GILDA/ UserTutorials]
[57] The SEPAC-Grid Project, http://www.sepac-grid.org
[58] LITBIO - [http://www.litbio.org]
[59] BioInfoGrid – [http://www.bioinfogrid.eu]
[60] The SwissBioGrid Project - [http://www.swissbiogrid.org]
[61] M. Podvinec, S. Maffioletti, P. Kunszt, K. Arnold, L. Cerutti, B. Nyffeler, R. Schlapbach, C. Türker, H. Stockinger, A.J. Thomas, M.C. Peitsch, T. Schwede, *The SwissBioGrid Project: Objectives, Preliminary Results and Lessons Learned.*, IEEE e-Science 2006, 2006, pp. 148-155.
[62] Laboratorio Italiano BioInformatica (LIBI) – [http://www.libi.it]

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Service-Based Access to and Processing of Large Scientific Datasets

Umit V. CATALYUREK [a,1], Sivaramakrishnan NARAYANAN [a], Olcay SERTEL [a], Jun KONG [a], B. Barla CAMBAZOGLU [a], Tony PAN [a], Ashish SHARMA [a], Shannon HASTINGS [a], Stephen LANGELLA [a], Scott OSTER [a], Tahsin KURC [a], Metin GURCAN [a] and Joel SALTZ [a]

[a] *Dept. of Biomedical Informatics, The Ohio State University, USA*

Abstract. Service oriented application development has gained wide acceptance in the Grid computing community. A number of projects and community efforts have been developing standards (OGSA and WSRF), tools, and middleware systems to support development, deployment, and execution of Grid Services. In this paper, we present the use of Grid Services in two application scenarios; servicing of large scale data from simulation-based oil reservoir management studies and computer-aided analysis of large microscopy images for neuroblastoma prognosis. We describe how the high-performance backend systems have been developed for these applications and how these backend systems have been exposed through Grid Service interfaces. Our work illustrates that with the help of appropriate middleware systems, it is possible to develop and deploy high-performance Grid Services that support storage, retrieval and processing of large scale scientific and biomedical datasets, while enabling remote access to the data via well-defined interfaces and protocols.

Keywords. Grid computing, grid service, data services, analytic services.

Introduction

There is an increasing number of collaborative research projects and applications, which involve access to heterogeneous and distributed collections of data and analytical resources, in all fields of science, engineering, and biomedical research. Many of these projects and applications require query and retrieval of large datasets and analysis of data through simple and complex operations. Simulation-based oil reservoir management studies [25,38] and analysis of digitized microscopy images [10,24] are two examples of applications that generate and reference large datasets (ranging from tens of Gigabytes to Terabytes in size). The datasets in these applications are usually captured and stored at different locations, prompting the need to provide support for remote access to datasets and/or remotely process them. The complexities of management of large data volumes and execution of compute intensive operations create many challenges when researchers would like to share their data and analysis applications. The challenges are compounded

[1]Corresponding Author: Umit V. Catalyurek, 3190 Graves Hall, 333 W10th Ave, Columbus, OH 43210, USA; E-mail: catalyurek.1@osu.edu

by the heterogeneous nature of data sources, in terms of dataset formats and support for query and retrieval, and of analytical resources, in terms of the format of input and output parameters of analysis programs and how these programs are invoked.

There has been considerable progress in Grid computing technologies and Grid-enabled middleware systems in recent years. Grid computing has moved from being an infrastructure to enable access to high-end computing systems at supercomputer centers towards becoming a platform to facilitate sharing and integration of distributed information, data analysis tools, and applications. Interoperability is a major concern in an environment where data and analytical resources are developed autonomously by different developers and when programmatic access to resources is required. To address the interoperability concerns, a services-based view of the Grid has emerged. In this view, data sources and applications are exposed to the environment using standard interfaces. Users interact with the resources through well-defined Grid services protocols. In this way, the complexities and heterogeneity of individual resources can be hidden from clients and greater interoperability among applications can be achieved.

Several core functions need to be supported in an end-to-end system for enabling data-driven scientific applications in a Grid environment. These functions include management of data types and metadata, virtualization of data sources, remote data analysis execution (including data product generation), and Grid services interfaces. We have developed a suite of middleware components and tools to support these functions. In this suite, DataCutter, which is a component-based middleware, enables combined use of task- and data-parallelism and is used to support data product generation (e.g., aggregates of data subsets) [7]; STORM [31,32] provides virtualization of file-based datasets as object-relational tables and support for data subsetting; Mobius [20] supports management of data definitions and data types as XML schemas, XML virtualization of data, and metadata management; Introduce [21] facilitates development and deployment of strongly-typed Grid services.

In the following section, we briefly describe two application scenarios from engineering and biomedical research. A short description of the middleware frameworks and tools is presented in Section 2. Sections 3 and 4 illustrate how these frameworks and tools are used to develop Grid-enabled data and analysis services in the two application scenarios.

1. Application Scenarios

1.1. Oil Reservoir Management Studies

Effective oil reservoir management requires accurate characterization of the reservoir properties and efficient management strategies that involve optimized placement of production and injection wells. Simulation-based oil reservoir management is a viable approach to evaluate different optimization strategies and to understand changes in reservoir properties over long periods of time [25]. Various production strategies (i.e., the number of placement of injection and production wells) applied to multiple realizations (simulation runs) of geostatistical models are simulated using a numerical model of the reservoir under study to incorporate geologic uncertainty. This approach can lead to large volumes of output data [38]. In addition, changes in reservoir characteristics (e.g., rock

properties) over time are tracked by seismic data simulations (or seismic measurements in the field). Data obtained from seismic and reservoir simulations are stored for analysis. The data analysis processes subsets of seismic simulation datasets and reservoir simulation datasets in order to generate summary data such as production rates over a time period, bypass oil regions in the reservoir, and rock properties in the reservoir. The results of the analysis can be used to refine the reservoir models, simulate new production strategies, and collect additional seismic data.

Simulations are performed on a three-dimensional mesh over several time steps. Each realization corresponds to different geostatistical models and different number of wells and well placements. At each time step, the value of seventeen separate variables and cell locations in 3-dimensional space are output for each cell in the grid. Common analysis scenarios involve queries for economic evaluation as well as technical evaluation, such as determination of representative realizations and identification of areas of bypassed oil. Examples of client requests include *"Find all the potential bypassed oil cells between time T_1 and T_2 in realization A"* and *"Retrieve the oil saturation values at all mesh points from realizations A and B between time steps T_1 and T_2; visualize the results."*

Seismic data is recorded as sound traces generated by multiple sound sources on the surface and sampled by receivers at the bottom and on the surface of the reservoir. The sound traces are used to infer subsurface material properties. The surveys can be either carried out in the field or simulated using the seismic models of the reservoir. A seismic dataset is stored in files in a standard exchange format, referred to as SEGY, defined by the Society of Exploration Geophysics. A seismic data file consists of a 3600-byte header followed by a record for each sound trace. Each record contains a 240-byte header and the sound trace. The header information stores the metadata associated with the sound trace including sound source id, receiver id, receiver location, the number of samples stored for the trace. Traces collected for a single sound source are usually stored in a single file. When numerical models are used to generate seismic data, each data file can be up to 25 Gigabytes in size and there can be thousands of data sources simulated, resulting in datasets ranging from a few terabytes to hundreds of terabytes in size.

Seismic data can be used in creating subsurface images and predicted subsurface material properties. The reservoir model can be revised by imaging and inversion of output from seismic data simulations. Imaging analysis requires that subsets of seismic data be selected based on, for example, the type of sensor in a recording array and for each or a suite of sources.

1.2. Computer-Aided Prognosis of Neuroblastoma

Neuroblastoma is a cancer that develops from sympathetic nervous system. The current prognosis of the disease is based on examinations of tissue samples by expert pathologists following the International Neuroblastoma Prognosis Classification System developed by Shimada *et al.* [39]. According to the International Classification System, each neuroblastoma case can be classified based on several morphological features such as neuroblastic differentiation (undifferentiated, poorly-differentiated, and differentiating subtypes), mitosis karyorrhexis index and the presence or absence of Schwannian stroma development (stroma-poor and stroma-rich tissue).

The manual examination of tissue samples by pathologists is not only time consuming but also subject to intra- and inter-reader variability. Therefore, it is crucial to

develop a computerized system that allows reproducible diagnosis. The system should also assists pathologists in their decision-making procedure. Recently, intensive research studies have been conducted to develop computer-aided prognosis methods. As a continuation of these efforts, a new computerized system is being developed for discriminating the grade of neuroblastic differentiation as well as analysis of Schwannian stromal development [18,23].

The input to this system is compressed, digital images of tumor tissue samples stained using haematoxylin and eosin. In this system, computer vision and pattern recognition methods are utilized in developing a novel multi-resolution classification system for discriminating the grade of neuroblastoma. A new segmentation method that merges the Fisher-Rao criterion into the generic Expectation-Maximization algorithm is proposed [23]. After extracting representative features from identified regions associated with different cytological components of interest, such as cytoplasm and neuropils, these features are used in the classification stage where multiple classifiers are employed. At each resolution level, a sequence of image processing steps, including segmentation, feature construction, feature selection, feature extraction, and classification, are applied. Furthermore, a decision rule that controls the transition process from the lower to higher resolution level is defined using a procedure that involves voting and weighting priori classifier accuracies over the training data.

As for the computerized analysis of Schwannian stromal development, a similar multi-resolution approach is employed, using texture analysis to classify each neuroblastoma image as stroma-poor or stroma-rich. The statistical texture features such as local binary pattern [34] features are extracted to construct the feature pool. Also, a feature selection step [37], which determines the most discriminating features, is employed.

In order to satisfy the requirements in terms of classification accuracy, due to large image sizes, the computational burden is excessive to be handled by a single computer. The performance gets worse when various components of the system for grade of differentiation and stroma classification are combined to provide the classification information that is required for prognosis. Therefore, it is essential to develop a powerful computation infrastructure for pathological image analysis, where the computation time can be significantly reduced.

2. Middleware and Development Tools

In this section, we briefly describe DataCutter, Mobius, STORM, and the Introduce toolkit. We have used these tools in the development of high-end backend systems and Grid services for the applications presented in Section 1.

2.1. DataCutter

DataCutter [7] is a component-based middleware framework [35,36,22,1,5,2,11] designed to support coarse-grain dataflow execution on heterogeneous environments. In DataCutter, application processing structure is implemented as a set of components, referred to as *filters*, which exchange data through *logical streams*. A *stream* denotes a unidirectional data flow from one filter (i.e., the producer) to another (i.e., the consumer). A filter is required to read data from its input streams and write data to its output streams

```
SELECT < Data Elements >
        FROM Dataset_1, Dataset_2, ..., Dataset_n
        WHERE < Expression > AND < Filter(< Data Element >) >
        GROUP-BY-PROCESSOR ComputeAttribute(< Data Element >)
```

Figure 1. Database queries supported by STORM.

only. The DataCutter runtime system supports both data- and task-parallelism. Processing, network and data copying overheads are minimized by the ability to place filters on different platforms. The filtering service of DataCutter performs all steps necessary to instantiate filters on the desired hosts, to connect all logical endpoints, and to call the filter's interface functions for processing work. Data exchange between two filters on the same host is carried out by memory copy operations while a message passing communication layer (e.g., TCP sockets or MPI) is used for communication between filters on different hosts.

2.2. STORM

STORM [31,32] is a services-oriented framework designed to support processing of large datasets in a distributed environment. It provides basic database support for 1) *selection of the data of interest* from scientific datasets stored in files and 2) *transfer of selected data from storage nodes to compute nodes for processing*. The current implementation is based on DataCutter middleware framework. Using the DataCutter runtime support, STORM can perform parallel I/O on distributed data and execute data selection and data filtering operations on heterogeneous collections of storage and compute clusters.

In order to support data subsetting on file-based datasets, STORM implements three abstractions: *virtual tables*, *select queries*, and *distributed arrays*. The first two abstractions are based on object-relational database models [40]. *SELECT* operation of the form shown in Figure 1 are supported on virtual tables. Data elements selected by the *SELECT* operation are grouped based on a computed attribute. In the figure, the $< Expression >$ statement can contain value-based selections and range queries. $Filter$ allows implementation of user-defined operations that are difficult to express with simple comparison operations.

The client program that carries out data processing can be a parallel program. In that case, the distribution among client nodes of the data elements returned as the result of the query can be represented as a *distributed array*. This abstraction is incorporated into the STORM framework by the *GROUP-BY-PROCESSOR* operation in the query formulation. $ComputeAttribute$ is another user-defined function that generates the attribute value on which the selected data elements are grouped together based on the application specific partitioning of data elements.

2.3. Mobius

Mobius is a middleware framework [29,26] designed for efficient metadata and data management in dynamic, distributed environments. It provides a set of generic services and

protocols to support distributed creation, versioning, management of database schemas, on-demand creation of databases, federation of existing databases, and querying of data in a distributed environment. Its services employ XML schemas to represent metadata definitions and XML documents to represent and exchange metadata instances. The role of *Global Model Exchange* (GME) service of Mobius is to ensure distributed model evolution and integrity while providing the ability for storage, retrieval, versioning, and discovery of models of all shape, complexity, and interconnectedness in a distributed environment. *Mobius Mako* is a service that exposes data resources as XML data services through a set of well-defined interfaces based on the Mako protocol. Our current Mako implementation provides support to expose XML databases that support the XMLDB API and contains an implementation of MakoDB. MakoDB is an XML database built on top of MySQL [30]. The Mako protocol defines methods for submitting, updating, removing, and retrieving XML documents. Upon submission, Mako assigns each entity a unique identifier. Documents or subsets of XML documents, can be retrieved by specifying their unique identifier. XML documents can be removed by specifying their unique identifier or by specifying an element identifying XPath [6] expression. XML documents that reside in a Mako can be updated using XUpdate[2].

2.4. Introduce

Introduce [21] is an open-source, extensible toolkit that supports easy development and deployment of Web Services Resource Framework compliant services. Introduce is designed to reduce the service development and deployment effort by hiding low-level details of the Globus Toolkit and to enable the implementation of strongly-typed services. In strongly-typed services, a service produces and consumes data types that are well-defined and published in the Grid. This enables data-level syntactic interoperability so that clients and services can access and consume data elements programmatically and correctly.

3. Grid Data Services for Datasets from Oil Reservoir Management Application

In Section 2.2, we introduced the STORM system that could answer SQL-like queries over scientific datasets distributed over multiple nodes in a cluster. A STORM *instance* may be setup to serve one or more of these datasets. A STORM *daemon* process is the front-end of an instance. It receives queries from a client and returns the results. It orchestrates the execution of the query by starting up tasks on nodes in the cluster. Frequently, however, datasets served by different STORM instances need to be correlated for analysis. For example, one STORM instance running in $cluster_0$ may serve reservoir simulation information while a different, remote STORM instance running in $cluster_1$ may serve seismic measurements. An analysis to determine a suitable location to drill for oil may require a combination of parts of these datasets. This necessitates the functionality of distributed query execution across STORM instances.

In this section, we describe the design and implementation of a layered infrastructure for serving large, distributed datasets using STORM and OGSA-DAI. The first layer in our infrastructure implements support for efficient use of distributed storage clusters

[2]http://www.xmldb.org

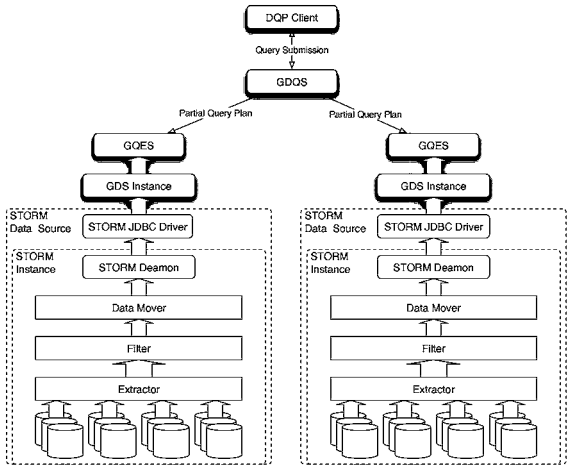


Figure 2. Querying multiple data sources using DQP, OGSA-DAI, and STORM.

and provides an object-relational virtualization of file-based datasets. This layer builds on the STORM middleware framework. The second layer leverages the existing work in the Grid community. The OGSA-DAI project[3] provides middleware for integrating and querying data from multiple data sources. To exploit their middleware, we expose a STORM instance as an OGSA-DAI data source. In Section 3.1, we briefly describe the OGSA-DAI infrastructure. In Section 3.2, we describe the integration of STORM with the OGSA-DAI infrastructure. We also provide experimental results using Oil Reservoir and Seismic Studies application data in Section 3.3.

3.1. OGSA-DAI

The Grid has emerged as an integrated infrastructure for distributed computation [13,15]. The Open Grid Services Architecture (OGSA) [14] defines mechanisms for creating, managing, and exchanging information among entities called Grid services. The objective of the OGSA-DAI initiative is to build upon the OGSA infrastructure to deliver high level data management functionality for the Grid. It defines services and interfaces that can be used by clients to specify operations on data resources and data.

The OGSA-DAI infrastructure consists of two middleware components: OGSA-DAI (Data Access and Integration) and OGSA-DQP (Distributed Query Processing). OGSA-DAI [33] is a middleware product that allows data resources, such as relational or XML databases, to be accessed via Grid services. An OGSA-DAI data service allows data to be queried, updated, transformed and delivered. OGSA-DAI thereby provides a means for users to Grid-enable their data resources. OGSA-DQP [4] is the distributed querying component of the infrastructure. It supports queries over OGSA-DAI data services and over other services available on the Grid, thereby combining data access with analysis.

To leverage this infrastructure, we exposed a STORM instance as an OGSA-DAI data source. The details of this implementation are described in Section 3.2.

[3] http://www.ogsadai.org.uk/

3.2. System Implementation

The Grid Data Service (GDS) is the central OGSA-DAI component. OGSA-DAI provides default implementation of a Grid Data Service (GDS) that can expose databases that implement the JDBC interface. In order to take advantage of this, we developed a JDBC driver implementation for STORM. This allows the default GDS implementation to use a standard interface to communicate with the STORM instance. The JDBC driver also exposes the metadata corresponding to the tables (virtual tables) that the particular STORM instance serves.

When the GDS receives a *SELECT* query, it passes it on to the JDBC driver. The JDBC driver parses the query into the internal format used in STORM and sends it to the STORM daemon over a TCP/IP connection. Results are forwarded from STORM to GDS via the JDBC driver, which implements the ResultSet interface. The JDBC driver reads the results from STORM as a stream of bytes. It then parses the results into appropriate Java objects. Note that parsing objects may also require conversion from little endian to big endian format.

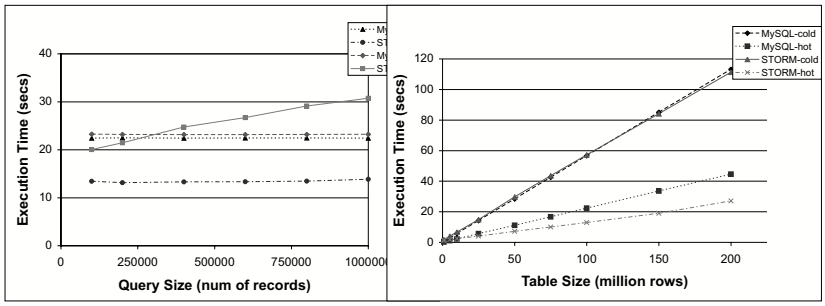
With this integration, as shown Figure 2, a STORM instance may be exposed as a data source. A user wanting to submit a query over disparate data sources that are wrapped by OGSA-DAI GDSs can also use the Distribute Query Processing (DQP) infrastructure, which implements a distributed query engine on OGSA-DAI data sources [4].

It is possible that for some kind of queries, the JDBC driver of STORM may prove to be a bottleneck. We incorporated a feature wherein the driver can interpret the data it receives from the STORM daemon in different ways. For example, an 84 byte record consisting of 21 floats may be interpreted as a single 84 byte array. The intuition behind this is that it is less expensive to interpret a record as an array of bytes rather than parse them into individual Java objects. This notion can be further extended to interpret a sequence of records as a single larger record to reduce the number of operations performed per record. We should note that this approach, however, will remove interoperability with OGSA-DQP since the DQP requires knowledge of individual attributes to execute operations like *PROJECT* and *JOIN*.

3.3. Experimental Results

For the experimental evaluation, we used two PC clusters. The first one, called *mob*, consists of 8 nodes equipped with dual 1.4 GHz AMD Opteron processors, 8 GB of memory and 1.5 TB of disk storage in RAID 5 SATA disk arrays. The nodes are connected to each other via a Gigabit switch. The second cluster, called *xio*, consists of 16 nodes, each node having two Intel Xeon 2.4GHz processors with hyper threading, resulting in 4 virtual CPUs per node. Each node has 4GB of memory and is connected to a distinct 7.3TB FAStT600 disk array. A detailed discussion of this cluster's I/O architecture can be found in [8]. We were able to achieve a raw I/O bandwidth (i.e., just reading data from disk without any selection operations) of 2.69 GB/s using STORM on the 16 *xio* nodes. Note that in the following experiments, we measure the efficiency of our integration upto the OGSA-DAI stage and not the DQP stage as overheads after the OGSA-DAI stage were out of our control.

In our first set of experiments, we carried out a preliminary performance comparison between STORM and MySQL and their OGSA-DAI implementations. OGSA-DAI



(a) (b)

Figure 3. (a) Comparison of MySQL and STORM. (b) Effect of large tables.

provides a default Data Resource implementation for MySQL. We generated tables with 6 floating point attributes, one of which is a unique attribute `idf`. No indexes were built on the tables. Experiments were done on a single node in the *mob* cluster.

In Figure 3(a), we compare the performance of MySQL and STORM for varying query sizes. Queries were of the form `SELECT * FROM T100m WHERE idf <` N where N corresponds exactly to the number of rows returned by the query. Here, `T100m` corresponds to a table with 100 million rows. We can see that STORM performs better than MySQL. Both systems take almost constant time to execute these queries because the scanning of the table takes up the bulk of the time. STORM's OGSA-DAI implementation is not as efficient as MySQL's, especially for very large queries. We attribute this to the fact that our current JDBC implementation is not well optimized. There is a high variable cost associated per tuple. However, STORM's implementation outperforms MySQL for queries smaller than 300,000 records. This is because the fixed (startup) cost associated with STORM-DAI implementation is lower than that of the MySQL-DAI implementation.

We should note that there are time and space overheads associated with importing file-based data into a database. In our experiments, we observed that when data is imported to MySQL using by piping the binary files via hexdump utility and using the *LOAD DATA INFILE* command, the transfer time was about 600 seconds per GB of data.

Figure 3(b) highlights the effect of growing table sizes on a query's performance. Queries were of the form: `SELECT * FROM T`X`m WHERE idf < 10000.0` where `X` is the size of the table in million rows. The cold-cache lines correspond to case when the filesystem's cache is purged by reading a large file. STORM and MySQL have almost identical performance with the cold-cache. Our results show that I/O costs dominate the execution time in the cold-cache case. The hot-cache experiments were done by repeating the queries several times to allow for benefits of the filesystem's caching. In these experiments, STORM has a 40% improvement over MySQL, which we attribute to the lower per tuple processing cost of STORM.

In the second set of experiments, we used a 315 GB size oil reservoir dataset. This data corresponds to the simulation output of a single realization as described in Section 1.1. For every time step, each point of three-dimensional grid is stored as a tuple of 21 attributes; the values of seventeen separate variables, cell locations in 3-dimensional space and realization id. Simulation is done over a grid of size $512 \times 512 \times 256$ for 60 time steps (Table 1). The dataset is partitioned across 3 data nodes such that each

Dataset	Attributes	Record Size (bytes)	Records (millions)	Dataset Size (GB)	Cluster Name Number of Nodes
Oil Reservoir	21	84	3,840	315	mob, 3
Seismic	16	4,240	247	1,056	xio, 16

Table 1. The characteristics of the datasets used in the experiments.

node has roughly 1/3 of the dataset. The dataset is indexed using an R-Tree [19] on the `X,Y,T,SOIL,VX` attributes where `X` and `Y` are spatial attributes, `T` refers to the time step, `SOIL` is the saturation of oil at the grid location and `VX` is the velocity in the X direction.

Figure 4(a) shows the timing results with varying query size. The queries involve retrieving all data in a rectangular region over increasing intervals of time. The number of records retrieved, thus, grows linearly. Since the dataset is indexed, there is almost no excessive I/O and STORM takes only about 2 seconds to execute these queries. The straightforward method of exposing this dataset, denoted by *STORM-DAI-o* is very inefficient, due to parsing of 21 attributes in every record and creation of excessive number of Java objects. Combining all 21 attributes into a single array (see Section 3.2), shown by the *STORM-DAI-1* line, results in a significant gain in performance. *STORM-DAI-50* combines 50 records into a single array which further improves performance. Combining larger number of records did not give us any further benefit. In Figure 4(a), *3 DAIs* shows the execution time when the dataset is distributed across three STORM-DAI-50 instances, each running on a single node. Each query was submitted to each STORM data source (i.e., the sub-tables) and the results were collected at the client. Partitioning the data across multiple data sources, improves parallelism among the STORM-DAI-50 instances resulting in better performance.

In the last set of experiments, we used 96 seismic data files compliant with the SEGY file format described in Section 1.1. Each file contained traces for a single sound source, generated by a simulation using a seismic model of the reservoir and was 11 GB in size. Files were evenly distributed across the 16 nodes of the *xio* cluster. The dataset was indexed using an R-Tree on the `RECV` attribute that corresponds to the receiver number for a particular trace.

On a single *xio* node, a base I/O rate of 193 MB/s was achieved using the *dd* Unix utility. As with any pipeline, bandwidth achieved increased with query size till a limiting value was reached. The bandwidth achieved at the end of the *Extractor* stage was 172 MB/s and at the end of the *Filtering* stage was 122 MB/s. The end-to-end bandwidth seen by the client was 55 MB/s. The memory-to-memory bandwidth, measured using the *memcpy* routine was about 560 MB/s. This was an important factor as communication between stages of the pipeline involves packing and unpacking tuples into/from buffers which involved a memory copy. Memory became the bottleneck that reduced the bandwidth at each stage of the pipeline. This is a problem in systems where the memory-to-memory bandwidth is comparable to I/O bandwidth.

Queries returned all records that involved a certain set of receivers denoted by a range: `SELECT * FROM segy WHERE RECV >= 1 AND RECV <=` X where we varied X linearly. The results of this experiment are shown in Figure 4(b). The number of records returned by such a query is $96 * X$.

We employed similar techniques as in the previous experiments to improve the query execution performance. For these queries, STORM performs the same amount of I/O

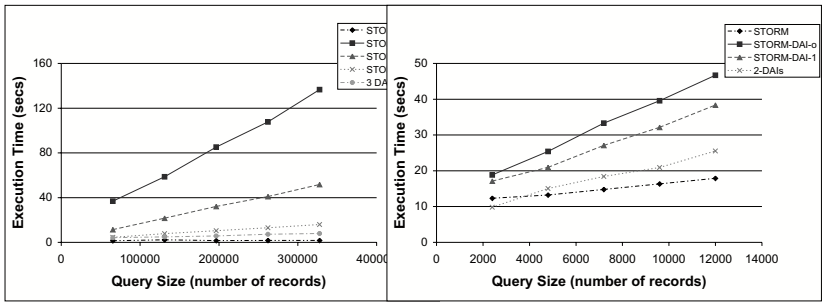


Figure 4. (a) Varying query sizes on oil reservoir simulation data. (b) Performance on Seismic Data.

and filters out unwanted tuples. The fixed cost is therefore STORM's startup costs, the constant I/O and filtering costs. The variable cost is that of transferring the result of the filtering operations. We can see that the naive OGSA-DAI implementation has a high variable cost. Treating records as an array of bytes reduces this cost. Combining several records into a single array, however did not cause any improvements unlike in the oil reservoir experiments. This is because of the small number of rows returned by the query. To further improve performance we set up STORM and GDS instances on two nodes to serve half the table (each) and noticed a further improvement in performance. This is reflected in the *2-DAIs* line in the figure.

4. Grid Analytical Services for Analysis of Neuroblastoma Images

In this section, we present a service-based [14] infrastructure that allows multiple, geographically distant researchers and software developers to access a common image data and code repository, letting them share their data while reusing the codes developed by the others. The infrastructure involves a multi-processor backend that enables fast parallel processing of images. It is specifically designed for very large-scale image processing and has pipelined processing capabilities. In this system, the images are automatically declustered by a central coordinator among multiple cluster nodes, where subimages are processed through Matlab [27], a high-level, commercial development environment, and the results (text or image) are collected back at the central coordinator. Furthermore, this backend is exposed as a Grid service and a Grid client is developed as a part of the system, for data management, remote image navigation, and job submission by end users.

Several recent papers address the use of Grid architectures [9,16,17,28,41,42] in biomedical applications. In [16], Giovanni *et al.* describe a service-based architecture, where various algorithms are provided as Grid services for drug design. In [28], Milanesi and Merelli describe their implementation of a high-performance Grid application for protein analysis. In [41], Sulakhe *et al.* present a scalable computational system for high-throughput analysis of genomes. In [42], Tirado-Ramos and Sloot focus on the design of a conceptual Grid architecture for interactive biomedical applications.

Our infrastructure differs from the above-mentioned works in that the parallel backend is based on a number of concurrently running high-level applications. Hence, the scientists can develop their algorithms using Matlab, which is a conventional and rapid development environment for most scientists. For a survey of related work on parallel Mat-

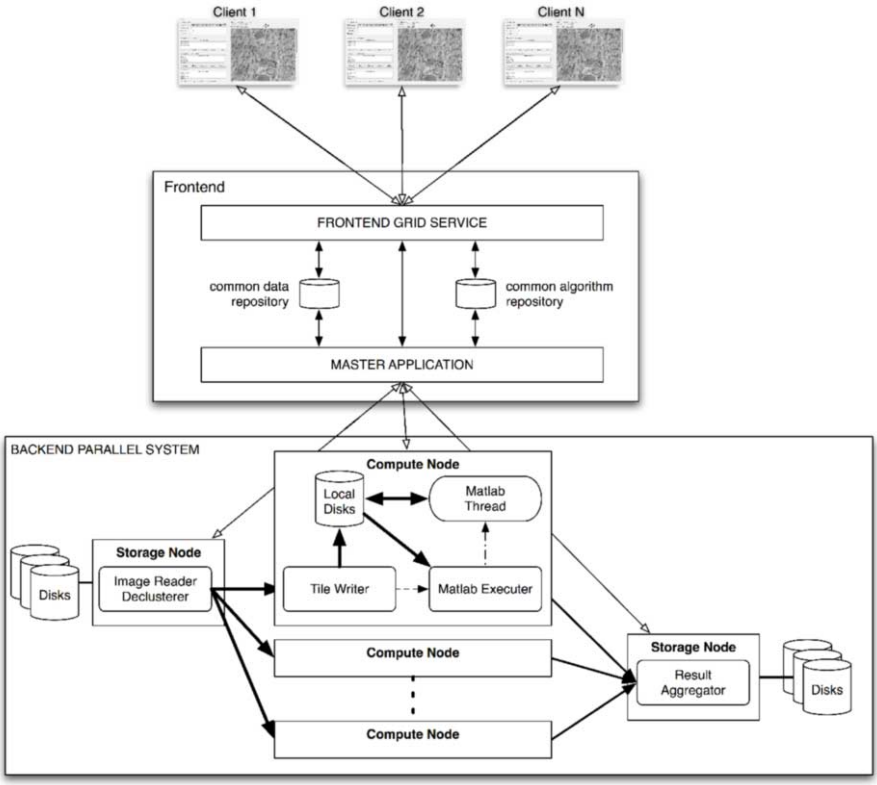


Figure 5. Overall system architecture.

lab environments, the reader is referred to [12]. Moreover, previous works focus on computational parallelism, whereas our emphasis in this work is on large-scale data-parallel processing, where the data needs to be stored on disk during the execution. Although our current prototype uses Matlab as the high-level algorithm development language for our image processing application, our design is generalizable and flexible enough to support other high-level languages such as IDL with minimal effort. Moreover, the developed architecture allows creation of simple workflows in the forms of pipelines of algorithms, i.e., the outputs of an algorithm can be fed as input to another, allowing quick integration of algorithms into a multi-stage image processing pipeline.

The general architecture of our framework includes a Grid client, a frontend Grid service, and a backend parallel system for processing very large images. The Grid client application contains a graphical user interface (GUI) for connecting to the frontend Grid service using a strongly-typed Grid service interface. Frontend provides the functionality to access a common algorithm and data repository and interacts with the backend processing system for executing image processing jobs. The backend system includes a master process that coordinates the data and task distribution on a number of storage and compute nodes of the backend system. The overall architecture with multiple Grid clients accessing the infrastructure is illustrated in Figure 5.

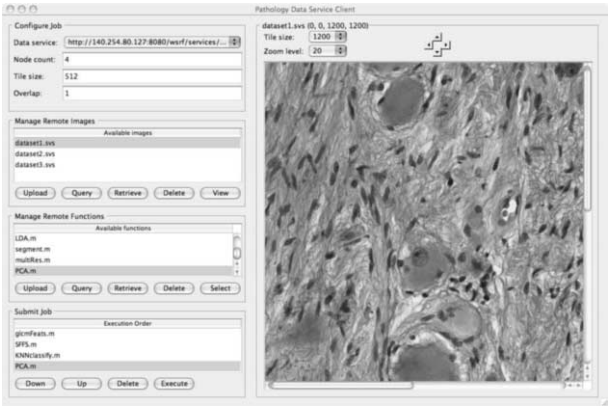


Figure 6. The Grid client GUI.

4.1. Client Application

The Grid client application, implemented as a Java application, contains a GUI for connecting to the frontend Grid service using a strongly-typed Grid service interface. A screen-shot of the client is provided in Figure 6. The Grid client provides functionality for

1. *managing the local and remote data and algorithm repositories* by providing some of the common operations found in a typical database management system. Users can upload their local data/algorithms to a remote Grid site, query the remote repository content, retrieve data and algorithms to the local disk, and delete the remote content.
2. *constructing simple imaging workflows in the form of pipelines* that will be executed on the parallel backend. The client also lets the user set several application-specific execution parameters. Here, we refer to a pipelined workflow combined with a specific set of execution parameters as a *job*.
3. *submitting jobs to the backend system* with the selected image data, selected set of algorithms, and configured parameters.
4. *visualizing the remote images* through the built-in image viewer. The viewer provides access to images in the data repository and functionality to remotely navigate on the images as well as zoom-in and zoom-out operations.

The Grid client is currently used during the development and evaluation of new algorithms. In the future, we plan to use it at the production level.

4.2. Grid Service

Grid service handles the interaction between the Grid clients and the backend parallel computing system. It provides methods for management (i.e., upload, query, retrieval, and deletion) of common image and algorithm repositories located on a master node. It is also responsible for configuring and initiating the image processing application at the backend in compliance with the job parameters set by the clients.

We developed the Grid service using the Introduce service development and deployment toolkit [21] as a strongly-typed service. Hence, client APIs and service interfaces

Table 2. Characteristics of the images used in the experiments

Slide Name	Resolution	Size (MB)	Class
Slide A	64990 × 59412	599	Schwannian stroma-poor
Slide B	75607 × 68443	1,700	Schwannian stroma-poor

operate on well-defined XML schemas. Due to the massive size of the images, we utilize the bulk data transfer for upload and retrieval of images. For this purpose, master node runs a gridFTP server [3] that facilitates transfer of the data over the Grid.

gridFTP is also used for transferring the portions of the images that are remotely visualized by the user. The service extracts the currently viewed portion of the image from the larger image on the disk and transfers it to the client via the bulk data transfer mechanism. Some simple caching mechanisms are implemented both at the server and client side to reduce the communication overhead while navigating in the images.

4.3. Parallel Backend

The parallel backend is developed using a version of the DataCutter middleware [7] that uses MPI as the underlying communication layer. The parallel backend runs a master application and a number of application daemons, implemented as a set of DataCutter filters. The master application is composed of a single reader filter, which reads the input images from the disk and divides them into image tiles. The image tiles are assigned to the application daemons running on the compute nodes, in a round-robin fashion. The tiles are transferred to the compute nodes and cached in their local disks.

Each application daemon is composed of two DataCutter filters: a tile writer filter and a Matlab executor filter. The writer filter is responsible for storing the incoming image tiles on the disk. For each tile, the writer passes the location of the image tile to a Matlab executor filter. The executor filter fires up a thread which runs Matlab at the very beginning. This thread runs the selected image processing applications through Matlab on the image tiles read from the disk. The results for each tile are written back to the disk, where they are fetched by the executor filter and passed to a result aggregator. After all image tiles are processed, the partial results are combined into the final result.

4.4. Experiments

We conducted experiments to evaluate the performance of the parallel image processing backend on a 16-node Linux cluster located at the Department of Biomedical Informatics at The Ohio State University. Each node of the cluster is equipped with dual 2.4 GHz Opteron 250 processors, 8 GB of RAM, and two 250 GB SATA drives providing 500 GB of local storage via software RAID0. The nodes are interconnected with switched gigabit Ethernet and Infiniband and are running Linux as the operating system.

We report various results using the algorithm we developed for analysis of Schwannian stromal development. Experiments are conducted on two neuroblastoma images of different sizes: a small image (Slide A) and a large image (Slide B). The properties of the images are given in Table 2. The images are divided into tiles of size 896 by 896 during the parallel processing.

Here, we report the performance results in terms of load imbalance, execution times, and speedups. The tests on four different whole-slide neuroblastoma images (in total,

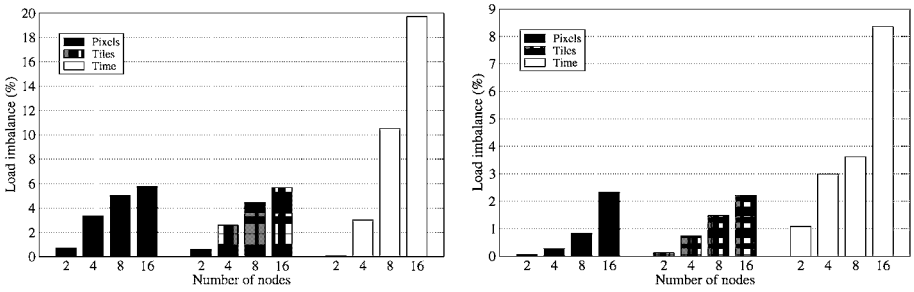


Figure 7. Load imbalance in terms of the number of pixels, the number of non-background tiles, and execution time with varying number of nodes.

images consist of 16,372 image tiles, each with a size of 896×896 pixels) show that the algorithm achieves a remarkable 96.55±2.33% classification accuracy in classifying neuroblastoma images into stroma-rich and stroma-poor regions.

Figure 7 presents the load imbalance over the parallel system during the classification for varying number of compute nodes using Slide A and Slide B. Here, the load imbalance is defined as $(W_{\max} - W_{\text{avg}})/W_{\text{avg}}$, where $W_{\max}$ and W_{avg} denote the load of the maximally loaded node and the average node load, respectively. The imbalances are displayed in terms of the number of pixels, the number of tiles, and actual execution time. Note that, in the algorithm, tiles having more than a certain percent of background pixels are considered as background tiles and not processed. Hence, the presented results are for the non-background tiles. According to Figure 7, round-robin distribution of tiles achieve a fair load balance. For both pixels and tiles, the imbalance is always under 6% for Slide A and under 3% for Slide B. On the other hand, the load imbalance in actual execution times is relatively higher. This is basically due to the multi-layered nature of the algorithm. The image tiles that the algorithm is unable to classify are tried to be re-classified at a higher resolution until a classification decision can be made on the tile. Consequently, some nodes process more tiles than the others, resulting in a load imbalance in the system. Since the classification decisions are made at the run time, no static load balancing strategy is possible and round-robin assignment is prone to load imbalance. It should also be noted that there is a difference between the load imbalances in processing the small and large images. This difference stems from the fact that the tile size is fixed while the image resolutions are different. Hence, the large image, which has the higher resolution, has more flexibility for load balancing. The tile size is fixed because the tile size of the training and test images must be the same.

Our analysis shows that using a single image reader is not sufficient as the number of compute nodes increases. This is because of the fact that the reader library we used is slow and becomes a bottleneck in providing the compute nodes with adequate number of image tiles. To alleviate this problem, we have employed a multi-reader approach that will guarantee that enough number of image tiles will be present for processing by the compute nodes. In this approach, multiple images are tiled concurrently by different readers running on different nodes and submitted to the compute nodes for processing. Figure 8 displays the execution times and the achieved speedups with varying number of compute nodes and two readers. As seen in the figure, it takes around 4.7 hours to

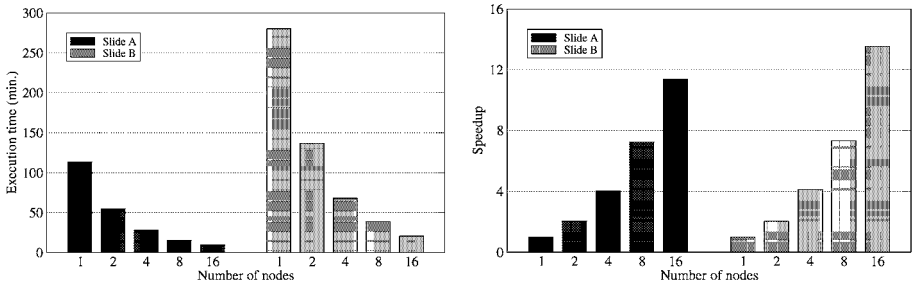


Figure 8. Execution times and speedups in processing small and large images with varying number of compute nodes.

classify the tiles of Slide B on a single-processor system. The execution time drops to about 21 minutes on a 16-node parallel system.

5. Summary

In this paper, we presented the use of Grid Services in two applications that involve access to large volumes of data and analysis of large datasets. In our implementations, Grid Service interfaces are coupled to high-end backend systems for storage and processing of data. The backend systems have been developed using DataCutter, which allows combined use of task and data parallelism on a cluster system, STORM, which implements high-performance query and data retrieval support for datasets stored in a set of distributed files, and Mobius, which provides XML-based support for metadata management. One of the applications have been exposed to the Grid environment using the OGSA-DAI framework, thus providing a OGSA compliant Grid Data Service. The service interfaces for the image analysis application, on the other hand, have been implemented using the Introduce toolkit and are compliant with the Web Services Resource Framework standards. Our work illustrates an example of combined use of Grid Service frameworks and high-performance middleware tools to create interoperable data and analytical resources that can handle large data and complex data analysis.

Acknowledgments

This research was supported in part by the National Science Foundation under Grants #CNS-0403342, #CNS-0615155, #CCF-0342615, and by NIH NIBIB BISTI #P20EB000591.

References

[1] The ABACUS project. *http://www.cs.cmu.edu/~amiri/abacus.html*.

[2] M. Aeschlimann, P. Dinda, J. Lopez, B. Lowekamp, L. Kallivokas, and D. O'Hallaron. Preliminary report on the design of a framework for distributed visualization. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 1833–1839, Las Vegas, NV, June 1999.

[3] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke. Gridftp protocol specification. *GGF GridFTP Working Group Document*, September 2002.

[4] M. Alpdemir, A. Mukherjee, A. Gounaris, N. W. Paton, P. Watson, and A. A. Fernandes. OGSA-DQP: A grid service for distributed querying on the grid. In *Proc. 9th International Conference on Extending Database Technology (EDBT)*, pages 858–861, 2004.

[5] K. Amiri, D. Petrou, G. R. Ganger, and G. A. Gibson. Dynamic function placement for data-intensive cluster computing. In *the USENIX Annual Technical Conference*, San Diego, CA, June 2000.

[6] A. Berglund, S. B. X. WG), D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. *XML Path Language (XPath)*. World Wide Web Consortium (W3C), 1st edition, August 2003.

[7] M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Distributed processing of very large datasets with DataCutter. *Parallel Computing*, 27(11):1457–1478, Oct. 2001.

[8] S. Bokhari, B. Rutt, P. Wyckoff, and P. Buerger. An evaluation of the osc fastt600 turbo storage pool. Technical Report OSUBMI_TR_2004_n02, The Ohio State University, Department of Biomedical Informatics, Sep 2004.

[9] V. Breton, C. Blanchet, Y. Lagre, L. Maigne, and J. Montagnat. Grid technology for biomedical applications. *Lecture Notes in Computer Science*, 3402:204–218, April 2005.

[10] U. Catalyurek, M. D. Beynon, C. Chang, T. Kurc, A. Sussman, and J. Saltz. The virtual microscope. *IEEE Transactions on Information Technology in BioMedicine*, 7(4):230–248, Dec 2003.

[11] Common Component Architecture Forum. *http://www.cca-forum.org*.

[12] R. Choy and A. Edelman. Parallel matlab: doing it right. *Proceedings of the IEEE*, 93(2):331–341, February 2005.

[13] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, USA, second edition, 2003.

[14] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The physiology of the Grid: An open grid services architecture for distributed systems integration. http://www.globus.org/research/papers/ogsa.pdf, 2002.

[15] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, Fall 2001.

[16] A. Giovanni, C. Massimo, F. Sandro, and M. Maria. Progengrid: a grid-enabled platform for bioinformatics. *Proceedings of HealthGrid 2005, from Grid to Healthgrid*, 112:113–126, 2005.

[17] M. N. Gurcan, T. Pan, A. Sharma, T. Kurc, S. Oster, S. Langella, S. Hastings, K. M. Siddiqui, E. L. Siegel, and J. Saltz. GridIMAGE: A novel use of grid computing to support interactive human and computer-assisted detection decision support. *Journal of Digital Imaging*, 20(2):160–171, 2007.

[18] M. N. Gurcan, T. Pan, H. Shimada, and J. Saltz. Image analysis for neuroblastoma classification: Segmentation of cell nuclei. *Engineering in Medicine and Biology Society, 28th Annual International Conference of the IEEE*, pages 4844–4847, August 2006.

[19] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of SIGMOD'84*, pages 47–57. ACM Press, May 1984.

[20] S. Hastings, S. Langella, S. Oster, and J. Saltz. Distributed data management and integration: The mobius project. In *GGF Semantic Grid Workshop 2004*, pages 20–38. GGF, June 2004.

[21] S. Hastings, S. Oster, S. Langella, D. Ervin, T. Kurc, and J. Saltz. Introduce: An open source toolkit for rapid development of strongly typed grid services. *Journal of Grid Computing*, March 2007.

[22] C. Isert and K. Schwan. ACDS: Adapting computational data streams for high performance. In *14th International Parallel & Distributed Processing Symposium (IPDPS 2000)*, pages 641–646, Cancun, Mexico, May 2000.

[23] J. Kong, H. Shimada, K. Boyer, J. Saltz, and M. Gurcan. Image analysis for automated assessment of grade of neuroblastic differentiation. *IEEE International Symposium on Biomedical Imaging, Accepted*, 2007.

[24] V. S. Kumar, B. Rutt, T. Kurc, U. Catalyurek, S. Chow, S. Lamont, M. Martone, and J. Saltz. Large image correction and warping in a cluster environment. In *Proceedings of SC2006 High Performance Computing, Networking, and Storage Conference*, 2006.

[25] T. Kurc, U. Catalyurek, X. Zhang, J. Saltz, R. Martino, M. Wheeler, M. Peszyńska, A. Sussman, C. Hansen, M. Sen, R. Seifoullaev, P. Stoffa, C. Torres-Verdin, and M. Parashar. A simulation and data analysis system for large scale, data-driven oil reservoir simulation studies. *Concurrency and Computation: Practice and Experience. To appear.*, 2005.

[26] S. Langella, S. Hastings, S. Oster, T. Kurc, U. Catalyurek, and J. Saltz. A distributed data management middleware for data-driven application systems. In *Proceedings of 2004 IEEE International Conference*

on Cluster Computing, September 2004.
[27] The Mathworks, Natick, MA. *Using Matlab*, 1997.
[28] L. Milanesi and I. Merelli. High performance grid based implementation for genomics and protein analysis. *Studies in Health Technology and Informatics*, 120:374–380, 2006.
[29] The Mobius Project. http://www.projectmobius.org.
[30] MySql Database. *http://www.mysql.com/*.
[31] S. Narayanan, U. Catalyurek, T. Kurc, X. Zhang, and J. Saltz. Applying database support for large scale data driven science in distributed environments. In *Proceedings of the Fourth International Workshop on Grid Computing (Grid 2003)*, pages 141–148, Phoenix, Arizona, Nov 2003.
[32] S. Narayanan, T. Kurc, U. Catalyurek, and J. Saltz. Database support for data-driven scientific applications in the grid. *Parallel Processing Letters*, 13(2):245–271, 2003.
[33] Open Grid Services Architecture Data Access and Integration (OGSA-DAI). http://www.ogsadai.org.uk.
[34] T. Ojala, M. Pietikainen, and M. T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.
[35] R. Oldfield and D. Kotz. Armada: A parallel file system for computational grids. In *Proceedings of CCGrid2001: IEEE International Symposium on Cluster Computing and the Grid*, Brisbane, Australia, May 2001. IEEE Computer Society Press.
[36] B. Plale and K. Schwan. dQUOB: Managing large data flows using dynamic embedded queries. In *IEEE International High Performance Distributed Computing (HPDC)*, August 2000.
[37] P. Pudil, J. Novovicova, and K. J. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.
[38] J. Saltz, U. Catalyurek, T. Kurc, M. Gray, S. Hastings, S. Langella, S. Narayanan, R. Martino, S. Bryant, M. Peszynska, M. Wheeler, A. Sussman, M. Beynon, C. Hansen, D. Stredney, , and D. Sessanna. Driving scientific applications by data in distributed environments. In *Dynamic Data Driven Application Systems Workshop, held jointly with ICCS 2003*, Melbourne, Australia, June 2003.
[39] H. Shimada, I. Ambros, L. Dehner, J. Hata, V. Joshi, and R. B. Terminology and morphologic criteria of neuroblastic tumors. *American Cancer Society*, 86(2):349–363, 1999.
[40] M. Stonebraker and P. Brown. *Object-Relational DBMSs, Tracking the Next Great Wave*. Morgan Kaufman Publishers, Inc., 1998.
[41] D. Sulakhe, A. Rodriguez, M. D'Souza, M. Wilde, V. Nefedova, I. Foster, and N. Maltsev. Gnare: automated system for high-throughput genome analysis with grid computational backend. *Journal of Clinical Monitoring and Computing*, 19(4-5):361–369, October 2005.
[42] A. Tirado-Ramos and P. M. A. Sloot. A conceptual grid architecture for interactive biomedical applications. *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, pages 762–767, 2006.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

A Feature-Rich Workflow Description Language that Supports Resource Co-allocations

Yonghong YAN and Barbara M. CHAPMAN
Department of Computer Science, University of Houston
{yanyh,chapman}@cs.uh.edu

Abstract. A very important topic in the effort of deploying workflow applications in grid environments is finding a means to describe the complex application structures that is simple for users yet expressive enough to provide the workflow scheduler with the information needed to make (good) scheduling decisions. A workflow description language defines syntax and semantics for specifying workflow tasks and their relationships; thus it provides an abstract and formalized representation of the complex workflow structures in text format. But the existing workflow description languages focus on expressiveness with respect to describing the data and control flow of workflow structures. They lack the ability to specify resource request information in support of resource allocations for workflow tasks by the scheduler. In addition, we have found that many of the features requested by end users are not, or are only partially supported in current workflow languages.

In this paper, we present a high-level abstract language for domain scientists to describe workflow applications in grid environments, the Grid Application Modeling and Description Language (GAMDL). GAMDL associates resource request information with a workflow description so that a workflow scheduler can make resource co-allocation requests based only on the workflow description itself. In terms of expressiveness, GAMDL is able to describe data-flow structures of complex domain problems, and also allows the definition of control-flow logic within the data-flow. Designed to be intuitive and suitable for users without a background in grid computing, GAMDL provides features that are not available in other languages.

Keywords. Workflow Description Language, Grid Environment, Workflow Scheduling, Resource Allocation, GRACCE, GAMDL

1. Introduction

Scientific workflow applications [1] are domain problems that make intensive use of numerical tools. They require high compute power for the simulation and visualization and entail the transfer, storage and analysis of a huge amount of data. A simulation of these problems in computational grids [2] incorporate multiple dependent modules to be executed in predefined order on multiple resources. An uninterrupted simulation requires the storage and transfer of module data between resources in a timely manner. Enabling such an application in grid environments is much more complex than enabling monolithic and single-executable applications.

A very important part of the effort of deploying workflow applications in grid environments is describing the complex application structures. A means to do so must be simple yet expressive enough to provide the workflow scheduler with the information it needs to make (good) scheduling decisions. A workflow description language defines syntax and semantics for specifying workflow tasks and their relationships; it provides an abstract and formalized representation of the complex workflow structures in text format. As the basis of the establishment and execution of grid workflows, workflow description languages are the widely-used meta-model to describe the physical processes of large-scale applications.

Existing workflow description languages focus on being expressive enough to describe the data flow and control flow of workflow structures. They do not provide syntax for specifying resource request information to support resource allocations for workflow tasks. For example, it is very difficult to specify such resource request information as the number of CPUs and the amount of memory of workflow tasks to support resource co-allocation and execution planning. One workaround could be to use another method, such as RSL [3], that is independent of the workflow description to make the resource multi-request. But this approach introduces complexities in reasoning about the resource multi-request based on the workflow task relationships. The workflow scheduler has to refer to two specifications to make resource allocation and scheduling decisions, one for resource allocation and one for workflow scheduling. In addition to support for scheduling, a workflow language should also provide ease of use, and should support workflows with a variety of different requirements. Some typical features include helping handle errors produced before, during and after task execution and dependency handling, support for partial workflow specification, and support for application-specific utilities for task launching, termination and restarting. These features are not, or are only partially supported in current workflow language development efforts.

In this paper, we present a high-level abstract language for domain scientists to describe workflow applications in grid environments, Grid Application Modeling and Description Languages (GAMDL). GAMDL associates resource request information with workflow description so that a workflow scheduler can make resource co-allocation request based only on the workflow description itself. GAMDL also allows job execution history to be specified in an application description, which may be utilized by the workflow scheduler for resource co-allocation and execution prediction. GAMDL is able to describe data-flow structures of complex domain problems, and also allows the definition of control-flow logic within data-flow. Intuitive and suitable for users without a background in grid computing, GAMDL provides many features that are not available in other languages.

The paper is organized as follows. In the next section, we discuss the background, related work and motivation for the design of GAMDL. Section 3 gives the main application entities specified using GAMDL. In Section 4, we show how an application, its data-flow and control-flow logic are described in GAMDL. In Section 5, we show how resource request information and execution history are specified. We then provide an example of a complex control-flow description in Section 6. Finally, we draw our conclusions in Section 7.

2. Background, Related Work and Motivations

The design of GAMDL was driven by the needs of a production application, the UH Air Quality Forecasting (AQF) application [4], and is the result of a collaboration between domain users and the authors to enable the AQF application on the University of Houston (UH) campus grid in the GRACCE project [4].

2.1. the AQF Scientific Workflow Application and the GRACCE Project

AQF is an integrated computational model for regional and local air quality forecasting, and is composed of three subsystems: the PSU/NCAR MM5 weather forecast model, the SMOKE emission system, and EPA's CMAQ chemical transport model. An AQF execution is a computational sequence of the three subsystems with increasing resolution and decreasing geographical boundaries. Figure 1 illustrates the workflow of a nested 2-day forecasting operation over a single region of interest by a three-domain computation. Each rectangle represents a workflow module (task) and each arrow indicates the flow of data between modules. The 36km domain computation provides coarse forecast data over the continental USA, the 12km domain provides data across the south central USA, and the 4km domain forecasts air quality across a smaller geographic region.

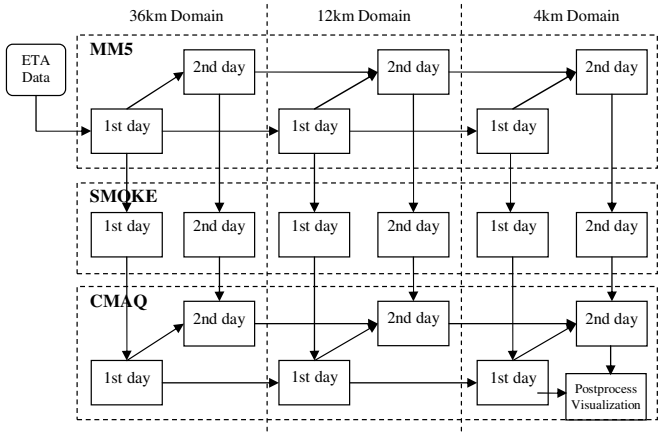


Figure 1. AQF Module Workflow

AQF modules are parallel MPI codes that require high performance parallel systems for their executions. A campus grid environment [5,6] that includes several Linux clusters with MPICH [7] as the MPI runtime has been set up for AQF daily forecasting. The AQF modules are installed on those cluster systems. In our original AQF workflow scheduler, AQF workflow structure is described using an XML file, and a Perl script is responsible for launching AQF workflow modules according to a predefined order. For each module, the resource are preallocated by the system administrators and is hard-coded in the XML file. The Perl script, which acts as the workflow scheduler, launches the tasks when the dependencies are resolved and initiates the transfer of intermediate files when these files are produced. Obviously, this type of human-scheduling policy is not suitable for changing grid environments and resource allocation should be automated to provide best possible decisions according to the resource load status.

GRACCE (Grid Application Coordination, Collaboration and Execution) project [8] was initiated to provide a general solution for managing scientific workflow applications in computational grid environments. The vision of GRACCE is to provide domain scientists with an integrated framework for building a custom grid application environment, from the management of the application and its dataset, to the automatic execution and viewing of results. In the GRACCE framework, end users are only required to provide descriptions of their applications and resource requirements. GRACCE is responsible for allocating grid resources for tasks in a workflow, placing tasks on resources for execution, monitoring them, and returning the results back to users as desired. The workflow scheduler developed in GRACCE addresses the resource allocation capability missing in our early XML/Perl solution and in most currently available workflow systems in computational grid environments. It features resource co-allocation and reservations, workflow execution planning, performance prediction and data/network-aware scheduling. Our initial simulation results show that the GRACCE workflow scheduler is able to improve the workflow performance by about 20% under a high resource load, compared to a regular workflow scheduler. We refer interested readers to [9,10] for more information about the project and the scheduler.

2.2. Application Modeling and Description: Related Works

During the process of deploying the AQF application, we were requested by users to provide a general-purpose and easy-to-use method to describe application entities and structures. Thus we needed a description language to support this and resource allocation in the GRACCE workflow scheduler.

2.2.1. Condor DAGMan

Condor [11] is a resource management system for distributed computing resources. The Directed Acyclic Graph Manager (DAGMan) [12] is a workflow scheduler for Condor jobs. DAGMan uses DAG as the data structure to represent job dependencies and the data-flow is specified by parent-child relationships. Condor does not have support for specifying the intermediate files between tasks, and users must specify data transfer through the preprocessing and postprocessing scripts associated with each job.

2.2.2. The Chimera Virtual Data Language

The Chimera Virtual Data System (VDS) [13] is a set of tools for data-processing workflow management. The workflow description language of Chimera is called Virtual Data Language (VDL) and it is a data-flow style language. In VDL, a set of application programs are described as *transformations* (TR) and the executions of transformations are described as *derivations* (DV). Derivations produce or consume data files, which are described as data objects. In Chimera, VDL definitions are stored in a catalog that provides for the tracking of the provenance of all files derived by an application. Chimera VDS contains the strategy for producing a given logical file. The dependencies between derivations in terms of data files constitutes the application abstract workflow in the form of a DAG of program execution steps.

2.2.3. Taverna XScufl

Taverna [14] is a workflow environment for grid life-science applications. Taverna uses an extended Scufl [15], XScufl, as the workflow specification language. The Scufl language is essentially a data-flow centric language. In Scufl, a logical service, an individual step within a workflow, is called a *processor*, which can be regarded as a function of some set of input data to a set of output data. A set of *data links* connect data source processors to data destination processors. Taverna has developed a set of processor plug-ins that handle the data flow on data links; for example, a WSDL Scufl processor implemented by a single Web Service operation described in a WSDL.

2.2.4. Askalon and Karajan

Askalon and Karajan workflow systems both define a control-flow style language to describe application logics [16,17]. In this style, the task execution order, which is implied by their dependency relationships, is explicitly specified using *sequential* and *parallel* syntactics. For example, given a 4-task diamond workflow, the specification could be "*{sequential A, {parallel B, C}, D}*". It is read: task *A*, task group *{parallel B, C}* and task *D* must be executed in sequential order, where tasks *B* and *C* can be executed concurrently. In this way, the dependency relationships between tasks are implicitly constrained by the task execution order that is specified by using the two syntactics. In addition to these two, Askalon and Karajan introduce other imperative programming structures, such as *for*, *if*, *switch*, etc., to specify complex control-flow logics. Karajan also allows the definition and use of variables and functions in the specification.

2.2.5. Triana

Triana [18] provides a graphical environment to enable the composition of workflow applications through mouse input. In the Triana workflow language, a component, the unit of execution, is a Java class with an identifying name, input and output "ports", a number of optional name/value parameters and a single process method. Triana uses both data-flow and control-flow in workflow description. In the case of data-flow, data arriving on the input "port" of the component triggers execution, and in the case of control-flow, a control command triggers the execution of the component. The execution of workflow within Triana is decentralized; data or control flow "messages" are sent along communication "pipes" from sender to receiver.

2.2.6. Others

YAWL [19] is a workflow language built upon two main concepts: workflow patterns and Petri net [20]. It was developed by taking Petri nets as a starting point and adding mechanisms to allow for more direct and intuitive support of different workflow patterns. Similar to the Karajan and Askalon approach, it is a control-flow style language.

Business Process Execution Language (BPEL) [21] is an XML-based workflow definition language to describe enterprise business processes in web services. In BPEL, a workflow step is described using WSDL. For scientific applications, either extensions to the language or the wrapping of the applications is needed to use BPEL.

Semantics web [22] standards, Resource Description Framework (RDF) [23] and Web Ontology Language (OWL) [24], aim to provide another structuring and descrip-

tion framework that allows data to be integrated in a much larger-scale than what current HTML-framework provides. The general-purpose semantic web standards are very abstract and additional vocabularies need to be defined for a specific field.

2.3. Discussions and Motivations

In reviewing those related efforts, we found that most existing workflow description languages focus on being expressive enough to describe the data flow and control flow of workflow structures. They lack the features for users to provide resource request information to support resource allocations for workflow tasks by the scheduler. For example, the resource request information for multiple dependent tasks could be specified to support workflow-orchestrated resource co-allocation and execution planning. One workaround is to make the resource multi-request using another method, such as RSL [3] that is independent of the workflow description. However, users have to derive the resource multi-request from the workflow task relationships and the derivation is a complex reasoning process. The workflow scheduler has to refer to two specifications, one for resource allocation and one for workflow scheduling, which complicates its decision making process during resource allocation and scheduling.

Aside from scheduling support, a workflow language should provide ease of use and should support workflows with a plethora of different requirements. Some typical features include helping handle errors produced before, during and after task executions and dependency handling, support for partial workflow specification, and support for application-specific utilities for task launching, termination and restarting. We have found that these features are not, or are only partially supported in the workflow language development efforts.

In terms of their structure, workflow languages provide for the description of workflow applications in either control-flow or data-flow style. In control-flow style, the execution order of workflow tasks are specified explicitly. It is either in the order of specification statements or specified using imperative-programming syntax, such as "sequential", "parallel", "fork" or "join". The formalized model for control-flow style is the Petri Net [20], and in [25,26], the authors show how Petri Nets are used for workflow description. Using data-flow style, a workflow description consists of the specifications of a set of tasks, one or multiple start tasks, and a set of dependency relationships between the parent and child tasks that share intermediate data. The dependency relationships determine the execution order of the workflow tasks. The data-flow description is the Directed Acyclic Graph (DAG) representation of the workflow and it is able to specify most of the current scientific workflow applications. Unlike the control-flow style, only dependency relationships between relevant tasks need to be specified. For big workflows, the control-flow style makes it very complex and error-prone for users to reason about the sequential or parallel execution orders of the tasks. But using data-flow style, the task execution order can be easily determined by the workflow scheduler based on the task dependency relationships. So the DAG-based data-flow style is a much easier method for users to describe static workflow than the Petri Net-based control-flow style.

Based on our study and the requirements of both AQF users and our workflow scheduler, we have designed a data-flow style workflow description language, Grid Application Modeling and Description Language (GAMDL). GAMDL provides supports for resource allocation to the workflow scheduler in great flexibility and extensibility by

allowing resource request details to be specified at different levels of abstraction. It is more powerful and flexible than the related efforts we have studied. We summarize the capabilities and features of GAMDL as follows:

- Supports the description of both data-flow and control-flow logic (loops and conditional branches) at a high level of abstraction.
- By separating the description of application logic and execution workflow, GAMDL supports the specification of partial workflows and reoccurring workflows without introducing additional complexities.
- GAMDL allows multiple jobs to be associated with a workflow module. The scheduler may choose the most suitable one according to the hardware and software environment of the allocated resource. For example, a workflow module has two binary codes, one for Intel X86 architecture and one for PowerPC architecture. GAMDL allows them to be specified as two jobs and lets the scheduler choose the right one based on the architecture of the resource allocated for this module.
- Supports the definition of nested or hierarchical workflows, i.e. a workflow contains another workflow.
- Associates the specification of resource requests and execution schedules with the module specifications to support resource co-allocation of workflow dependent modules. When allocating resources for a module, the scheduler evaluates resources based on not only the module itself, but also on its parent, child and sibling modules. For example, given a parent module and a child module, the scheduler considers allocating the same resource for them so that there is no need to transfer the intermediate files on the network.
- Allows for the specification of application-specific scripts for various purposes, such as preprocessing, postprocessing, checking the state of job execution, cleaning temporary file, killing jobs, etc.
- Allows similar modules to be easily described using multi-value properties. The description document is structured by using entity uid and uid references. These two features greatly improve the usability of the language and the readability of the description document.

GAMDL is XML language-based and the GAMDL syntax is developed as a set of XML-Schema [27]. XML is the most widely used modeling language for workflow description in grid computing and has a very rich set of development tools support. XML-Schema is used to define a set of rules to which an XML document must conform in order to be considered 'valid'. It is a W3C standard and provides a rich data model that allows us to express sophisticated structures and constraints used in GAMDL. The use of XML-Schema for GAMDL helps us easily develop a GAMDL parser using the open source XML development library, Apache XMLBeans [28]. XMLBeans binds XML data with Java objects through the schema of the data expressed in XML-Schema. In our example, after we design the GAMDL in XML-Schema, the XMLBeans compiler takes the GAMDL schema and generates Java codes that access a GAMDL document. All the data types, XML documents and elements in GAMDL are mapped to Java classes. Using these automatically generated codes, we can easily develop a GAMDL parser in pure Java language.

3. GAMDL Entities and Core Concepts

As a data-flow style language, GAMDL describes a workflow application by specifying the application entities and the relationships and dependencies between entities. In this section, we discuss the main application entities used in GAMDL.

3.1. GAMDL Execution Specification

In distributed and heterogeneous environments, a computational job is often specified in an abstract, platform-independent format by users and runtime systems translate the specification to a platform-dependent launching script that is used to create the execution instance. The main issue in developing a specification language for different level of users and runtime is how to flexibly define the properties of an execution in such an extent that different level of users can specify them in the abstraction level only they need. To support such flexibility, GAMDL splits an execution specification into two parts, the execution schedule and the execution configuration. An execution schedule includes information about *when and where* the executable is launched, such as its start time and host name. An execution configuration includes information about *how* the executable is launched, such as its launcher (e.g. a bash shell), directory, arguments and environment variables. In this separation, the platform-independent information in the execution configuration is supplied by users. When the scheduler allocates a resource for the execution, its fills in the execution schedule. Based on the architecture of the allocated resource, the runtime system fills in the platform-dependent part of the execution configuration. Another information that is part of an execution specification is the resource request. The resource request lists the resource details required for this execution, such as the number of CPU and the memory size. Based on this information, the scheduler makes resource allocation decisions and generates an execution schedule. An execution is of ExeType type in GAMDL schema, as shown in the following code fragment.

```
<xsd:complexType name="ExeType">
  <xsd:sequence>
    <xsd:element name="executable" type="xsd:string"/>
    <xsd:element name="location" type="xsd:anyURI"/>
    <xsd:element name="version" type="gamdl:VersionType"/>
    <xsd:element name="exeType" type="gamdl:ExecutableTypeEnumeration"/>

    <xsd:element name="exeSchedule" type="gamdl:ExeScheduleType"/>
    <xsd:element name="exeConfig" type="gamdl:ExeConfigType"/>
    <xsd:element name="resrcReq" type="gamdl:ResourceReqType"/>

    <xsd:element name="supportArch" type="gamdl:SysArchType"/>
    <xsd:element name="requiredLibrary" type="gamdl:LibraryType"/>
    <xsd:element name="docPage" type="xsd:anyURI"/>
  </xsd:sequence>
  <xsd:attribute name="uid" type="gamdl:UidType"/>
</xsd:complexType>
```

ExeType Schema

In the ExeType schema, most of the fields (elements or attributes in XML terms), such as executable and location elements, are self explainable. The execution schedule is specified in the exeSchedule field of ExeScheduleType type, the execution configuration in the exeConfig field of ExeConfigType type and the resource request in the resrcReq field of ResourceReqType type.

```
<xsd:complexType name="ExeScheduleType">
  <xsd:attribute name="startTime" type="xsd:dateTime"/>
  <xsd:attribute name="host" type="xsd:string"/>
  <xsd:attribute name="hostCPUList" type="xsd:string"/>
  <xsd:attribute name="numCPU" type="xsd:integer"/>
  <xsd:attribute name="memSize" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="retry" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ResourceReqType">
  <xsd:attribute name="startTime" type="xsd:dateTime"/>
  <xsd:attribute name="endTime" type="xsd:dateTime"/>
  <xsd:attribute name="maxNumCPU" type="xsd:integer"/>
  <xsd:attribute name="minNumCPU" type="xsd:integer"/>
  <xsd:attribute name="maxWTime" type="xsd:long"/>
  <xsd:attribute name="maxCPUTime" type="xsd:long"/>
  <xsd:attribute name="maxMem" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="minMem" type="xsd:nonNegativeInteger"/>
</xsd:complexType>
```

ExeScheduleType and ResourceReqType Schemas

The above code fragment shows the schemas of ExeScheduleType and ResourceReqType. The retry field in ExeScheduleType is used to specify how the system should rerun this schedule if the last execution fails. The retry string is defined in format of "[Integer]:[Integer]:[Integer][+|x|e]". The first integer is the maximum number of retries; the second one is the first interval (in second) and the base to calculate the interval between each retry; The last [Integer][+|x|e] is used to define how the interval is calculated in each retry, e.g. 4+ means the next interval is the current one plus 4; 2x means the next interval is 2 times the current one; 2e means that next interval is the current one powered by 2. For example, "retry="5:2:2x"" means that the scheduler should retry this execution up to 5 times if the first one fails, and the intervals between each try, in seconds, are 2, 4, 8, 16, 32.

```
<xsd:complexType name="ExeConfigType">
  <xsd:sequence>
    <xsd:element name="launcher" type="xsd:string"/>
    <xsd:element name="launcherArgu" type="xsd:string"/>
    <xsd:element name="directory" type="xsd:string"/>
    <xsd:element name="arguments" type="xsd:string"/>
    <xsd:element name="env" type="gamdl:EnvironmentType"/>
    <xsd:element name="stdin" type="xsd:string"/>
    <xsd:element name="stdout" type="xsd:string"/>
    <xsd:element name="stderr" type="xsd:string"/>

    <xsd:element name="validator" type="gamdl:ScriptType"/>
    <xsd:element name="preprocesser" type="gamdl:ScriptType"/>
    <xsd:element name="postprocesser" type="gamdl:ScriptType"/>
    <xsd:element name="cleaner" type="gamdl:ScriptType"/>
    <xsd:element name="killer" type="gamdl:ScriptType"/>
    <xsd:element name="doctor" type="gamdl:ScriptType"/>
  </xsd:sequence>
</xsd:complexType>
```

ExeConfigType Schema

In the ExeConfigType type, the six fields of ScriptType type are used to specify the application-specific scripts for different purposes. The validator script is used to check whether a completed execution generates the expected results or not; for example whether the data in the output files are correct or not. While the execution's exit code from the operating system tells whether it has completed or failed, the use of the validator script allows users to validate the execution using application-specific methods or algorithms. The preprocesser script and the postprocesser script are launched before

the executable is launched and after the execution is complete. The cleaner script is used to clean out the temporary files after execution. The killer is the script used to kill the execution process and its child processes if it has any. This is very useful for terminating all the processes of an MPI program. The doctor script is used to check the health of the execution and it is often called by the run time system.

Using these six scripts and the retry feature mentioned before, users can design robust and automated failure detection and restart functions for an execution. For example, if a parallel MPI application runs much longer than its past executions, it is very likely that the application's parallel processes have lost the state of internal communications and it hangs forever. The runtime system detects this using the provided doctor script and kills the hanging processes using the killer script. It then validates whether the expected results are generated using the validator script because the application may have lost communication state at the end of the execution, e.g. when closing the communication sockets after all the application data are generated. If it is a failed execution, the runtime system invokes the cleaner script to clean the temporary and incomplete output. According to the retry pattern configured in the retry string, the runtime system restarts the execution. These features are requested by a group of application users [4] and are very useful for applications that run frequently, e.g. daily. Users do not need to intervene very often to deal with those errors that can be recovered from by the system.

3.2. Module and Job Specification

The core entity in GAMDL for describing a workflow application is "module". A module is an application component to accomplish certain application goals, typically, processing input files and generating the required output files. A module is associated with one or more jobs and their executions are all able to accomplish the module's goals. A common case of having multiple jobs is when the module code has been compiled into several binaries for different platforms. Each of these binaries can be specified in one job. A module may have multiple input and output files. All the module jobs should consume these same input files and generate the same output files. The schema of the GAMDL module, the ModuleType, is shown in the following.

```
<xsd:complexType name="ModuleType">
  <xsd:sequence>
    <xsd:element name="inputFiles" type="gamdl:UidRefSetType"/>
    <xsd:element name="outputFiles" type="gamdl:UidRefSetType"/>
    <xsd:element name="jobSpec" type="gamdl:ModuleJobSpecType"/>
    <xsd:element name="dftExeConfig" type="gamdl:ExeConfigType" />
    <xsd:element name="dftExeSchedule" type="gamdl:ExeScheduleType"/>
    <xsd:element name="dftResrcReq" type="gamdl:ResourceReqType"/>
    <xsd:element name="metaJobSpec" type="gamdl:MetaJobSpecType"/>
  </xsd:sequence>
  <xsd:attribute name="uid" type="gamdl:UidType"/>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="dftJobIndex" type="xsd:integer"/>
</xsd:complexType>
```

ModuleType Schema

The inputFiles and the outputFiles fields in the schema are self-explanatory. The jobSpec and the metaJobSpec fields are for specifying the two types of GAMDL module jobs, the regular job and the meta-job. A regular job is a single execution and it is of ModuleJobSpecType type. A meta-job, of MetaJobSpecType type, is a workflow. The use of a meta-job allows the construction of nested workflows where a workflow

contains another workflow. In a MetaJobSpecType-typed meta-job, the workflow is specified by the appRun field of AppRunType type. We shall discuss this schema later on. The schemas for the two types of jobs are shown in the following.

```
<xsd:complexType name="ModuleJobSpecType">
  <xsd:sequence>
    <xsd:element name="exeSchedule" type="gamdl:ExeScheduleType"/>
    <xsd:element name="exeConfig" type="gamdl:ExeConfigType"/>
    <xsd:element name="resrcReq" type="gamdl:ResourceReqType"/>
  </xsd:sequence>
  <xsd:attribute name="index" type="xsd:integer" use="required"/>
  <xsd:attribute name="uid" type="gamdl:UidType" use="optional"/>
  <xsd:attribute name="name" type="xsd:string" use="optional"/>
  <xsd:attribute name="description" type="xsd:string" use="optional"/>
  <xsd:attribute name="exeSpecUid" type="gamdl:UidType"/>
</xsd:complexType>

<xsd:complexType name="MetaJobSpecType">
  <xsd:sequence>
    <xsd:element name="argu" type="gamdl:VariableType"/>
    <xsd:element name="appRun" type="gamdl:AppRunType" />
    <xsd:element name="appRunFile" type="xsd:anyURI" />
  </xsd:sequence>
  <xsd:attribute name="index" type="xsd:integer" use="required"/>
  <xsd:attribute name="uid" type="gamdl:UidType" use="optional"/>
  <xsd:attribute name="appRunUid" type="gamdl:UidType" use="required"/>
  <xsd:attribute name="name" type="xsd:string" use="optional"/>
  <xsd:attribute name="description" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Schema for Module Job Specification

In the ModuleJobType schema, the exeSpecUid attribute references an ExeType-typed execution that is already defined. The three elements, the exeSchedule, the exeConfig and the resrcReq, are for specifying the job-specific execution details and resource request. Although the execution specification referenced by the exeSpecUid attribute also provides these, allowing them to be provided here gives users the option of supplying module-specific details for an execution, and allows them to customize an execution in one module without changing the execution itself. For the same reason, the dftExeConfig, dftExeSchedule and dftResrcReq fields in the ModuleType schema are for specifying the corresponding default for jobs. If a field is not given in the job specification, the corresponding default is used. Also, since a module may belong to one or more applications or workflows, these fields may be given in the application and workflow description too, in order to provide custom execution details specifically for that application or the workflow.

Finally, we note that both regular and meta module jobs can be identified via an index, the index field of the above schemas. An index is a sequence number that orders the job specifications. The dftJobIndex field in the ModuleType schema tells the scheduler which job should be considered first when allocating resources.

3.3. Multi-Value Property and Entity Uid

In applications, such as AQF, multiple modules use the same executable with different, but similar execution details and input/output files. For example, there are 6 different CMAQ modules in the AQF workflow shown in Figure 1 for forecasting air quality in three domains and for two days. The differences in the use of the six modules are in the specification of the execution configuration and input/output file names. To describing them one by one is tedious work. Moreover, changes in the specification of the CMAQ execution may requires changes in that of all the six CMAQ modules. This has proved to

be an error-prone editing process and the readability of the resulting description is poor. We have introduced additional language support for these to improve the usability and readability of workflow specifications. These new features in GAMDL are provided via the "multi-value property" and the "entity unique id".

A **Multi-value property** (mvproperty), as its name implies, is a property that may have multiple values. It is defined as $mvpName = \{v0, v1, ..., vn\}$, and is referenced by $\$mvpName$. $\#mvpName$ denotes the number of values defined. A reference to $mvpName$ replicates the referencing sentence $\#mvpName$ times; in each replica, the reference is replaced with a distinct one of its values. For example, if we define $dmsz = \{36k, 12k, 4k\}$, and $day = \{d1, d2\}$, the sentence aqf-$mm5$-$\${dmsz}$-$\${day}$ represents all six instances ($\#dmsz * \#day$) of the AQF MM5 modules in Figure 1. In an XML document, the replication of an mvproperty reference is on an element basis. When the GAMDL parser encounters a reference to an mvproperty, it replicates the nearest outer element that contains the reference. This element is called the containing element of the mvproperty reference. The parser does not recursively process the same references in the child element of the containing element, instead, it instantiates all references to the mvproperty in a replicate element with the same value.

In the GAMDL description for the AQF application, the mvproperties are defined as following:

```
md={mm5, smoke, cmaq}    # Three AQF subsystems
dmsz={36k, 12k, 4k}      # Three AQF domains
day={d1, d2}             # Two-day forecasting
vdmsz={12k, 4k}          # The visualized domain
```

The following code fragment describes the six CMAQ modules in Figure 1 using these mvproperties.

```
<module uid="cmaq-${dmsz}-${day}">
  <inputFiles>
    <ref uid="cmaq-${dmsz}-${day}-in1"/>
    <ref uid="cmaq-${dmsz}-${day}-in2"/></inputFiles>
  <outputFiles>
    <ref uid="cmaq-${dmsz}-${day}-out1"/>
    <ref uid="cmaq-${dmsz}-${day}-out2"/></outputFiles>
  <jobSpec name="cmaq-${dmsz}-${day} job spec">
    ... ...
  </jobSpec>
</module>
```

CMAQ Module Definition

The **entity unique id (uid)** is an attribute to uniquely identify an entity, for example a job or a module, as shown in their schemas presented above. It is provided by users when specifying an entity. To include a defined entity in another entity's specification, the user only needs to reference it by its uid. In the above code fragment for CMAQ module definition, the module uid is defined to be "cmaq-${dmsz}-${day}" which covers the six modules. The input/output files are specified by the uid references of the corresponding files. This is the typical usage of uids in GAMDL, that is, the application entities are defined in several documents, one for each type of entity; and the documents for specifying entity relationships and workflows use these entities via their uid reference. In this way, changes in the entity specification do not necessitate changes in the documents for higher-level application specification. It enables the creation of well-organized document structures that match human approaches to organizing this information. The use

of uids also enables the re-use defined entities in other applications; it can be used as the key or foreign key reference when storing an entity in an RDBMS or XML database.

4. Application and Workflow Specifications

GAMDL models a domain problem as an application and an execution of the application as a workflow. It allows both data-flow and control-flow logics to be described using data-flow style syntax.

4.1. GAMDL Application and Workflow

In GAMDL, application and workflow are two different concepts. An application is a high-level model of a domain problem from the viewpoint of an end user; a workflow is an execution instance of the application. An application definition includes the specifications of all the application entities and of the entity relationships and dependencies. A workflow definition specifies which application entities are included in the workflow execution and with which module(s) the execution starts; the runtime workflow is then constructed based on the high-level application specification. One advantage of this separation is that it allows users to specify multiple and different workflows of an application based on their needs without defining a new application each time. The support for subworkflow and partial workflow is thus provided naturally from this separation. A GAMDL application is defined in an application document and a GAMDL workflow in an appRun document. The structure of the application document is illustrated in the following code fragment.

```
<application name="" uid="" description="">
  <version ... />
  <appExes>
    <ref uid="" />
      ... ...
  </appExes>
  <appDataFiles>
    <ref uid="" />
      ... ...
  </appDataFiles>
  <appModules>
    <ref uid="" />
      ... ...
  </appModules>
  <appMdRships>
    <pCnRshipSet>
      ... ...
    </pCnRshipSet>
  </appMdRships>
</application>
```

GAMDL Application Document Structure

As this code fragment demonstrates, the collection of the definition of all application entities is via uid references. These references are organized in three child elements, appExes, appDataFiles, and appModules, each for one of the three types of application entities respectively, i.e. execution, file, and module. The descriptions of module relationships and dependencies are in the appMdRships element: we discuss its details below. The next code fragment shows the *AppRunType* schema for the appRun document.

```
<xsd:complexType name="AppRunType">
  <xsd:sequence>
    <xsd:element name="description" type="xsd:string"/>
    <xsd:element name="dftMdJobIndex" type="xsd:integer"/>
    <xsd:element name="dftMdJobResrcReq" type="gamdl:ResourceReqType"/>
    <xsd:element name="dftMdJobExeConfig" type="gamdl:ExeConfigType"/>
    <xsd:element name="dftMdJobExeSchedule" type="gamdl:ExeScheduleType"/>

    <xsd:element name="mdRun" type="gamdl:MdInAppRunType"/>
    <xsd:element name="startMd" type="gamdl:UidRefSetType"/>
        ... ...
  </xsd:sequence>
  <xsd:attribute name="uid" type="gamdl:UidType"/>
  <xsd:attribute name="appUid" type="gamdl:UidType"/>
  <xsd:attribute name="startTime" type="xsd:string"/>
        ... ...
</xsd:complexType>

<xsd:complexType name="MdInAppRunType">
  <xsd:sequence>
    <xsd:element name="jobResrcReq" type="gamdl:ResourceReqType"/>
    <xsd:element name="jobExeConfig" type="gamdl:ExeConfigType"/>
    <xsd:element name="jobExeSchedule" type="gamdl:ExeScheduleType"/>
  </xsd:sequence>
  <xsd:attribute name="ref" type="gamdl:UidType"/>
  <xsd:attribute name="jobIndex" type="xsd:integer"/>
</xsd:complexType>
```

AppRunType

In the workflow schema, the application specification is referenced via the appUid attribute. The startTime string states when the workflow should be launched: it allows a sophisticated cron job format for reoccurring executions of the workflow. The mdRun element of MdInAppRunType type is for specifying the inclusion of application modules. It includes the module uid reference, the job index of the module, the execution details, i.e. the execution configuration and the execution schedule, and the resource request. This schema allows users to provide workflow-specific execution details and a module job index. As the ModuleType schema, the three elements, dftMdJobResrcReq, dftMdJobExeConfig and dftMdJobExeSchedule, are used to specify the default values of resource request and execution details of the workflow modules. These default values are used for a module whenever its mdRun element does not provide them. Lastly, the starting module(s) of the workflow are specified in the startMd element as uid references to the workflow modules. The following code fragment is an example of an AQF workflow definition.

```
<appRun uid="uhaqfrun" appUid="uhaqf" startTime="2005-07-16T15:23:15">
  <mvproperty file="uhaqf.mvproperties"/>
  <modules>
    <ref uid="eta-download"/>
    <ref uid="mm5-${dmsz}-${day}"/>
    <ref uid="smoke-${dmsz}-${day}"/>
    <ref uid="cmaq-${dmsz}-${day}"/>
    <ref uid="postv-${vdmsz}-${day}"/></modules>
  <startMd><ref uid="eta-download"/></startMd>
</appRun>
```

An AQF Workflow Definition

In the workflow specification, users do not need to specify the module relationships and dependencies. They are all given in the appMdRships element of the application document. In the following two subsections, we explain how data-flow and control-flow logics are described in GAMDL.

4.2. Application Data-flow Description

GAMDL models the application data-flow as a DAG, and captures both the dependency relationships between modules and the intermediate files associated with these relationships. A dependency relationship can be defined in either parent-children (PCn) pattern or child-parents (CPs) pattern. A PCn relationship, specified as a PCnRship element in GAMDL, has a parent module and one or more child modules, and a CPs relationship, as a GAMDL CPsRship element, has a child module and one or more parent modules. Intermediate files in a relationship are specified as pipes, one file per a pipe. A **pipe** has a pipeIn element and a pipeOut element; the pipeIn element represents the piped output file of the parent module, and the pipeOut element the piped input file of the child module. The next code fragment is part of the appMdRships element in the AQF application document. It describes the PCn relationships between the SMOKE and CMAQ modules, and between the CMAQ modules for the first day and the second day forecasting of the AQF application in Figure 1.

```
<appMdRships>
  <mvproperty file="uhaqf.mvproperties"/>
    ... ...
  <PCnRship parentMdUidRef="smoke-${dmsz}-${day}">
    <childMd uidRef="cmaq-${dmsz}-${day}">
      <viaPipe pipeIn="smoke-${dmsz}-${day}-out1"
        pipeOut="cmaq-${dmsz}-${day}-in1"/>
    </childMd>
  </PCnRship>

  <PCnRship parentMdUidRef="cmaq-${dmsz}-d1">
    <childMd uidRef="cmaq-${dmsz}-d2">
      <viaPipe pipeIn="cmaq-${dmsz}-d1-out1"
        pipeOut="cmaq-${dmsz}-d2-in1"/>
    </childMd>
  </PCnRship>
    ... ...
</appMdRships>
```

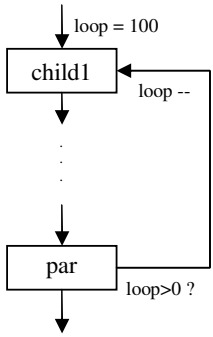
PCn Relationship Examples

4.3. Control-Flow Logic Description

GAMDL allows the specification of control-flow logics, such as loops or conditional branches, by introducing conditional pipes and variables. A **conditional pipe** associates a pipe with a boolean *if* condition which will be evaluated after the module completes execution. If it evaluates to *true*, the pipe is processed; otherwise, it is not. If the conditions of all pipes in a relationship are evaluated to *false*, runtime dependency is not established and the child module will not be executed. A **GAMDL variable** is a <*name*, *value*> pair associated with an *if* condition. A new value can only be assigned to the variable if the associated *if* condition evaluates to *true*; if no *if* condition is specified, an assignment is always made. If the *value* being assigned is in the form of *value1:value2*, *value1* is assigned if the *if* condition is *true* and *value2* is assigned otherwise.

In GAMDL, complex flow controls are achieved by the proper assignment of variable values and reasoning on the conditions associated with pipes and variables. The run time system can assign values to variables before a module's execution (in a preAssign element) and/or after a module's execution (in a postAssign element). The condition associated with a variable assignment or a pipe is permitted to reference system environment variables as well as GAMDL variables defined in other modules. In the following

for loop example, the *child1* module postAssigns 100 to the loop index (*loop*) if the *loop* variable has not yet been defined (which means this iteration is entering the loop), or *${loop} - 1* in each subsequent iteration. In the *pCnRship* of *par* module and *child1* module, a null pipe (using */dev/null* file) is specified with an *if* condition as *${loop} > 0*. In each iteration, if the condition evaluates to "*true*", the pipe is established and control passes to the *child1* module.




```
<module uid="child1">
  ... ...
  <postAssign name="loop" value="${loop}-1:100"
    if="defined(loop)"/>
</module>

<PCnRship parentMdUidRef="par">
  <childMd uidRef="child1">
    <viaPipe inFileUidRef="/dev/null"
      outFileUidRef="/dev/null" if = "${loop} > 0"/>
  </childMd>
</PCnRship>
```

For Loop

In Section 6, we give a more detailed example showing how a workflow with loops and conditional branches is specified using conditional pipes and variables. We want to note here that the control-flow specification introduces additional complexity in reasoning about the module execution order and is best used for the specification of simple control-flow logics.

5. Support for Resource Co-Allocations

GAMDL introduces the specification of two types of information about a module to aid a workflow scheduler when it is making resource allocation decisions; the resource request information of a module or workflow job, and the historical and profiling information about a job's execution on a resource.

5.1. Resource Request Specification

As we mentioned before, the resource request of an execution is specified using the ResourceReqType type shown below. It lists the resource details required by the execution. Two attributes we want to note here are the host and hostCPUList. These two fields are optional. If they are specified, it tells the scheduler that dedicated resources are requested, otherwise the scheduler allocates resources for the execution and fills in these two fields. This allows users to manually allocate resources for some workflow tasks and specify them in the execution specification. Using those dedicated allocations as hints, the workflow scheduler can make much better decisions for tasks for which users do not specify dedicated resources.

```
<xsd:complexType name="ResourceReqType">
  <xsd:attribute name="startTime" type="xsd:dateTime"/>
  <xsd:attribute name="endTime" type="xsd:dateTime"/>
  <xsd:attribute name="host" type="xsd:string"/>
  <xsd:attribute name="hostCPUList" type="xsd:string"/>
  <xsd:attribute name="maxCPU" type="xsd:integer"/>
  <xsd:attribute name="minCPU" type="xsd:integer"/>
  <xsd:attribute name="maxWTime" type="xsd:long"/>
  <xsd:attribute name="maxCPUTime" type="xsd:long"/>
  <xsd:attribute name="maxMem" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="minMem" type="xsd:nonNegativeInteger"/>
</xsd:complexType>
```

ResourceReqType

GAMDL associates the resource specification with a workflow module, which provides a natural solution for specifying resource multi-requests according to the workflow. When allocating resources for module jobs, a scheduler makes resource allocation decisions based on the module dependency relationships, for example, sibling module jobs are allocated concurrent resources if possible. The scheduling process is thus orchestrated by the workflow. Under Globus RSL [3], users have to explicitly specify the resource multi-requests for the purpose of resource co-allocation [29,30]. These resource multi-requests have to be constructed manually according to the dependency relationships of the workflow modules. While it would be possible to associate the RSL of module jobs with the workflow, the ability to specify resource multi-requests in RSL is then lost.

5.2. Job Execution Profile

The **Execution Profiles** of a module job in GAMDL are the historical and profiling information about the job executions on different grid resources. They provide the scheduler the historical information on observed executions in order to predict its future execution. For applications like AQF that run every day under similar scenarios, it is very easy to predict the execution of a module job on the resource on which it has been executing. Based on such prediction, the scheduler can make much better resource co-allocation decisions for module jobs. Also statistical analysis, data normalization and scaling may be performed on the historical data for other purposes, such as improving resource usage.

The GAMDL schema for a job execution profile is ExeProfileType, shown in the following. GAMDL organizes the execution profiles of a module job based on the resources the job has run on, with one profile for each resource. A module job may have been executed several times on a resource; each execution is described as an execution scenario, consisting of a list of the resources consumed, of ExeScenarioType type. In each profile, scaling algorithms are defined to calculate the possible resource usage based on the available scenarios.

```
<xsd:complexType name="ExeProfileType">
  <xsd:sequence>
    <xsd:element name="name"/>
    <xsd:element name="resourceName"/>
    <xsd:element name="scenarios"
                 type="gamdl:ExeScenarioType"/>
    <xsd:element name="CPUTimeScalingAlgorithm"
                 type="gamdl:ScalingAlgorithmType"/>
    <xsd:element name="VMemScalingAlgorithm"
```

```
                type="gamdl:ScalingAlgorithmType"/>
            ...
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ExeScenarioType">
 <xsd:sequence>
  <xsd:element name="name" type="xsd:string"/>
  <xsd:element name="numberOfCPU"/>
 <xsd:element name="hostCPUList"/>
  <xsd:element name="startTime"/>
  <xsd:element name="endTime"/>
  <xsd:element name="cpuTime"/>
  <xsd:element name="wallTime"/>
  <xsd:element name="memSize"/>
  <xsd:element name="swapSize"/>
  <xsd:element name="vmSize"/>
   ...
 </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ScalingAlgoNameType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="LINEAR"/>
   <xsd:enumeration value="INVERSELINEAR"/>
   <xsd:enumeration value="SEQURE"/>
   <xsd:enumeration value="EXPONENTIAL"/>
  </xsd:restriction>
</xsd:simpleType>
```

Job Execution Profile Specification

6. A GAMDL Example for A Workflow With Loops and Conditional Branches

In this section, we use a workflow example in Figure 2 to show how complex control-flow logic, such as loops and conditional branches, is described in GAMDL. In the example workflow, the module *md2* generates different output files (*F1*, *F2* or others) in different loops and these files are processed by module *md3*, *md4* or *md5*, respectively. The loop count is 100.

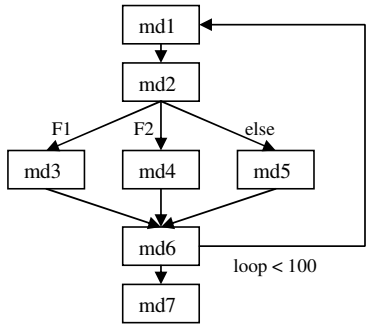


Figure 2. A Workflow with Loop and Conditional Branches

In the GAMDL description shown below, module *md1* postAssigns a *loop* variable, whose initial value is *100* and stride is *-1*. The module *md2* postAssigns two variables, *F1recent* and *F2recent*. *F1recent* is set to *true* if file *F1* is generated by *md2* in the last execution, otherwise *F1recent* is set to *false*; *F2recent* is handled similarly with respect to file *F2*. The pipe condition for *md3-md2* CPsRship is set to "*pipe(F1)* && *${F1recent}*", which is evaluated to *true* if *F1* is generated in the last execution and is available for

piping in. The *if* conditions for the *F2* pipe in *md4-md2* CPsRship and the else-pipe in *md5-md2* CPsRship are similar to the *F1* pipe. Loop control is specified in *md1-md6* CPsRship of *md1* and *md6* using a null pipe with condition "${*loop*}<*100* ".

In this example, GAMDL uses condition functions, such as *generated(F1)*, in a condition string. A **condition function** is a regular function (binary or script) that returns a boolean value; it should not make any modification to its externals. In the following specification, the *pipe(fileName)* function checks whether a file can be piped in or not. The *generated(fileName)* function checks whether the module execution generates the specified file; the *defined(variableName)* function checks whether a variable is defined or not.

```
<application name="LoopCon Example" uid="loopcon" ...>
  <appModules>
    <module uid="md1">
      <postAssign name="loop" value="${loop}-1:100" if="defined(loop)"/>
    </module>
    <module uid="md2">
      <outputFiles>
        <ref uid="F1"/>
        <ref uid="F2"/>
        <ref uid="Fx"/></outputFiles>
      <postAssign name="F1recent" value="true:false" if="generated(F1)"/>
      <postAssign name="F2recent" value="true:false" if="generated(F2)"/>
    </module>
    <module uid="md3"><inputFiles><ref uid="F1"/></inputFiles></module>
    <module uid="md4"><inputFiles><ref uid="F2"/></inputFiles></module>
    ...
  </appModules>

  <appMdRships>
    <cPsRshipSet>
      <CPsRship childMdUidRef="md2">
        <parentMd uidRef="md1">
          <viaPipe> ... </viaPipe></parentMd></CPsRship>
      <CPsRship childMdUidRef="md3">            <!--md3-md2 CPsRship -->
        <parentMd uidRef="md2">
          <viaPipe if="pipe(F1) && ${F1recent}"
            inFileUidRef="F1" outFileUidRef="F1"/></parentMd></CPsRship>

      <CPsRship childMdUidRef="md4">            <!--md4-md2 CPsRship -->
        <parentMd uidRef="md2">
          <viaPipe if="pipe(F2) && ${F2recent}"
            inFileUidRef="F2" outFileUidRef="F2"/></parentMd></CPsRship>

      <CPsRship childMdUidRef="md5">            <!--md5-md2 CPsRship -->
        <parentMd uidRef="md2">
          <viaPipe if="!{F1recent} && !{F2recent}"
            inFileUidRef="Fx" outFileUidRef="Fx"/></parentMd></CPsRship>

      <mvproperty name="md345">
        <value>md3</value>
        <value>md4</value>
        <value>md5</value></mvproperty>

      <CPsRship childMdUidRef="md6">
        <parentMd uidRef="${md345}">
          <viaPipe if="" inFileUidRef="${md345}-out"
            outFileUidRef="${md345}-out"/></parentMd></CPsRship>

      <CPsRship childMdUidRef="md1">            <!--md1-md6 CPsRship -->
        <parentMd uidRef="md6">
          <viaPipe if=" ${loop} < 100" inFileUidRef="/dev/null"
            outFileUidRef="/dev/null"/></parentMd></CPsRship>

      <CPsRship childMdUidRef="md7">
        <parentMd uidRef="md6"><viaPipe ... /></parentMd></CPsRship>
    </cPsRshipSet>
  </appMdRships>
</application>
```

7. Conclusions

In this paper, we presented the Grid Application Modeling and Description Language (GAMDL), a high level abstract language for domain users to describe a workflow application. GAMDL, designed to address the limitations of current workflow languages with respect to support for resource co-allocations for workflow tasks, is feature rich. It associates resource request specification and execution profile with workflow module specification, so resource multi-requests can be easily constructed by software based on the workflow. The execution details of a workflow job can be specified in different contexts, thus providing a flexible means to organize and reuse the entities in different applications without having to redefine them. Using GAMDL, both data-flow and control-flow relationships can be described using DAG style structures. Users do not need to manually construct the application's control-flow if they have a data-flow application. Other features of GAMDL include the use of mvproperties to describe similar entities, and its support for nested and partial workflow.

Acknowledgements

We wish to thank Dr. Daewon Byun and his team for supporting our efforts to explore AQF and for discussing the requirements of GAMDL. The work has been performed as part of the University of Houston's Sun Microsystems Center of Excellence in Geosciences.

References

[1] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Workflows for e-Science: Scientific Workflows for Grids*. Springer, 2006, ch. 1.

[2] I. Foster and C. Kesselman, "Computational Grids," in *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, CA, USA: Springer, 1999, ch. 2.

[3] "The Globus Resource Specification Language RSL v1.0," http://www-fp.globus.org/gram/rsl_spec1.html.

[4] B. Chapman, P. Raghunath, B. Sundaram, and Y. Yan, "Air Quality Prediction in a Production Quality Grid Environment," in *Engineering the Grid: Status and Perspective*, B. D. Martino, J. Dongarra, A. Hoisie, L. T. Yang, and H. Zima, Eds. American Scientific Publishers.

[5] Y. Yan, B. M. Chapman, and B. Sundaram, "Air Quality Forecasting on Campus Grid Environment," in *the Workshop on Grid Applications: From Early Adopters to Mainstream Users, GGF14*, June 2005.

[6] B. M. Chapman, H. Donepudi, Y. Li, P. Raghunath, B. Sundaram, Y. Yan, and J. He, "An OGSI-Compliant Portal for Campus Grids," in *10th ISPE International Conference on Concurrent Engineering: Research and Applications*, 2003, pp. 987–994.

[7] "MPICH," http://www-unix.mcs.anl.gov/mpi/mpich/index.htm.

[8] Y. Yan and B. Chapman, "Campus Grids Meet Applications: Modeling, Metascheduling and Integration," *Journal of Grid Computing*, vol. 4, no. 2, pp. 159–175, June 2006.

[9] "GRACCE: GRid Application Coordination, Collaboration and Execution," http://www.cs.uh.edu/~gracce.

[10] Y. Yan, "PhD Dissertation: Scheduling Scientific Workflow Applications in Computational Grids," Department of Computer Science, University of Houston, Tech. Rep., April 2007.

[11] M. Litzkow, M. Livny, and M. Mutka, "Condor - A Hunter of Idle Workstations," in *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[12] "DAGMan (Directed Acyclic Graph Manager)," http://www.cs.wisc.edu/condor/dagman.

[13] I. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation," in *14th International Conference on Scientific Database Management*, 2002, pp. 37–46.

[14] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: Lessons in Creating a Workflow Environment for the Life Sciences," *Concurrency and Computation: Practice & Experience*, vol. 18, no. 10, pp. 1067–1100, 2006.

[15] "XScufl Language Reference," http://taverna.sourceforge.net/docs/xscuflspecification.html.

[16] M. Wieczorek, R. Prodan, and T. Fahringer, "Scheduling of Scientific Workflows in the ASKALON Grid Environment," *ACM SIGMOD Record Journal*, vol. 34, no. 3, pp. 56–62, 2005.

[17] "Java CoG Kit Karajan/Gridant Workflow Guide," http://www.cogkit.org/release/4_0_a1/manual/workflow.pdf.

[18] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang, "Programming Scientific and Distributed Workflow with Triana Services," *Concurrency and Computation: Practice & Experience*, vol. 18, no. 10, pp. 1021–1037, 2006.

[19] W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: Yet Another Workflow Language," *Information System*, vol. 30, no. 4, pp. 245–275, 2005.

[20] T. Murata, "Petri Nets: Properties, Analysis and Applications," in *Proceedings of the IEEE*, vol. 77, Apr. 1989, pp. 541–580.

[21] "BPEL4WS: Business Process Execution Language for Web Services v1.0," http://www.106.ibm.com/developerworks/webservices/library/wsbpel.

[22] D. D. Roure and J. A. Hendler, "E-Science: The Grid and the Semantic Web," *IEEE Intelligent Systems*, vol. 19, no. 1, pp. 65–71, 2004.

[23] "Resource Description Framework (RDF)," http://www.w3.org/RDF.

[24] "OWL Web Ontology Language Overview," http://www.w3.org/TR/owl-features.

[25] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management," *The Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[26] N. R. Adam, V. Atluri, and W. Huang, "Modeling and Analysis of Workflows Using Petri Nets," *Journal of Intelligent Information Systems*, vol. 10, no. 2, pp. 131–158, 1998.

[27] "W3C XML Schema," http://www.w3.org/XML/Schema.

[28] "The XMLBeans," http://xmlbeans.apache.org/.

[29] K. Czajkowski, I. T. Foster, N. T. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A Resource Management Architecture for Metacomputing Systems," in *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*. London, UK: Springer-Verlag, 1998, pp. 62–82.

[30] K. Czajkowski, I. Foster, and C. Kesselman, "Resource Co-Allocation in Computational Grids," in *Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 1999, p. 37.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Parallelization and Scalability of Multiplayer Online Games via State Replication

Jens MÜLLER-IDEN [a], Sergei GORLATCH [a] and Tobias SCHRÖTER [b]
[a] *University of Münster, Germany*
{*jmueller*|*gorlatch*}*@math.uni-muenster.de*
[b] *University of Braunschweig, Germany*

Abstract. Massively Multiplayer Online Games (MMOGs) are an increasingly popular class of real-time interactive distributed applications that require scalable network architectures and parallelization approaches. While games of the role-playing genre already allow thousands of users to concurrently participate in a single game session, there are important genres, in particular action and strategy games, which have not been scaled to the massively multiplayer realm so far. These games have hard requirements in terms of scalability, in particular regarding density, because many players tend to congregate in small locations in these games. In this paper, we provide a comprehensive analysis of different scalability dimensions for online games and describe our novel approach of game state replication for scaling the density of players. The practical impact of our work is demonstrated by porting the popular action game QFusion, based on the famous Quake 2, onto our proxy-server system architecture using the replication approach. We discuss our solutions for maintaining data consistency of distributed replicas of the game state, as well as our enhancement of the game implementation in order to support multi-server replication. The experiments with the ported QFusion demonstrate its high responsiveness and show that our approach allows to almost triple the maximum number of players on four servers as compared with a single-server version.

Keywords. Virtual Environments, Scalability, Parallelization, Replication, Multiplayer Online Games

Introduction

Multiplayer online games played over the Internet have become a very popular class of distributed applications and are played by a continuously increasing number of users worldwide. Technically, an online game application is a distributed system consisting of several user clients and servers; a single application instance operates in real-time on a common application state replicated at all participating processes. The application updates the state frequently, such that users get the impression of a continuously evolving virtual environment.

In the area of distributed interactive real-time systems, online games are the most demanding applications in terms of responsiveness and scalability. In a running distributed game session, there are two main functionalities carried out by participating processes:

(1) the *server component* manages and regularly updates the application state depending on user inputs and the application-specific logic; (2) the *client component* regularly visualizes the updated application state to the user and forwards user inputs to the server component. Different actual placements of these components in the distributed architecture are possible; they directly determine important characteristics of the distributed application, like the achievable responsiveness and scalability. As an architecture solution for online games, the the client-server and the peer-to-peer approaches have been mostly used. Besides these well-known approaches, more sophisticated multi-server architectures for online games have been proposed by academia [5,7] and implemented in commercial game titles. The main goal of these recent approaches is to provide high scalability of the architecture to support high user numbers in a single game session, thus allowing to operate *Massively Multiplayer Online Games (MMOG)*. While traditional multiplayer game types, also known as *game genres*, like *First Person Shooter (FPS)* or *Real-Time Strategy (RTS)* usually are played with a few dozens of players, MMOGs, especially the sub-genre of *Massively Multiplayer Online Role Playing Games (MMORPG)*, are played with possibly several thousands of concurrent users in a single distributed application instance. Such high player numbers are possible, because the virtual game world of MMORPGs is huge and thus can be subdivided into independent regions, so-called *zones*, for concurrent parallel processing. Unfortunately, the virtual game worlds of FPS and RTS games are usually much smaller, such that the zoning approach is much less feasible for these types of games. Consequently, FPS and RTS games usually can still only be played in small-scale sessions.

Generally speaking, interactive online applications and distributed virtual environments all share the continuous real-time computation scheme. However, particular technical characteristics are quite different among the various types of applications. In terms of scalability, which is most important for the trend of making applications massively multi-user, several different *scalability dimensions* have to be distinguished by characterizing which particular application characteristic should be scalable. Some characteristics like the size of the virtual world or the total number of users can be scaled quite well already with known mechanisms. However, other characteristics of virtual environments are barely scalable yet. Especially the density of interactable objects, the so-called *entities* is very important for a wide range of interactive applications and game types as described and motivated further in this paper. These entities actually constitute the dynamic and modifiable state of a virtual environment: increasing their density is an urgent requirement for providing a highly dynamic and interactive application.

The overall size of a virtual world, as an example of a scalability dimension, can be quite easily scaled using the described zoning approach by adding additional zones to increase the overall world size. In contrast, the density of interactable entities, i.e., the number of those entities in a region of a fixed size in the environment, is difficult to scale and at best marginally supported by existing parallelization approaches: increasing the density requires the distributed architecture to increase the processing power dedicated to a particular region of the virtual world, which raises hard challenges in terms of consistency and correctness of the application processing. Efficient solutions for these challenges are not obvious and not available so far. However, this entities' density scalability dimension is crucially important for a wide range of virtual environment applications and game types. Such virtual environments can not be scaled properly yet using currently well-known scalability mechanisms.

This paper summarizes our novel *virtual world replication* approach as a suitable parallelization concept for scaling the density of interactable objects in virtual environments and online real-time applications. Scaling the density of these entities allows a whole range of application types to be run in a massively multi-user manner for the first time. For example, the presented replication approach allows FPS and RTS online games to be played with a substantially increased number of concurrent players. In order to operate the replication concept in a responsive manner, we discuss our *Proxy-Server Architecture* as a distributed architecture suitable to run the replication concept on multiple servers. Instead of subdividing the virtual environment into independent zones, the novel replication approach replicates all relevant data at participating server components. This way, the replication does not rely on particular characteristics of the environment and, therefore, can be applied to arbitrary types of interactive real-time applications. In contrast to existing approaches, the novel replication approach does scale the density of interactable entities in a virtual environment, because the overall processing of the application data is now distributed and parallelized among several server-components, such that each component only has to process a particular data subset. In order to keep the data consistent between distributed server components, several different synchronization mechanism analyzed, resulting in the use of an adapted lazy primary copy replication concept guaranteeing *eventual consistency* as a good compromise for the trade-off between responsiveness and level of consistency of the distributed real-time applications.

The contribution of this paper is threefold: (1) we provide a comprehensive analysis of the scalability dimensions required by different genres of massively multiplayer online games (Sections 2 and 3); (2) we present and discuss our replication parallelization approach to scaling FPS and RTS games using our *Proxy-Server Architecture* [10,9] as an alternative to conventional architectures (Sections 3 and 4); (3) as the main contribution, we demonstrate the practical impact of our replication approach for the important and large class of FPS games by porting the open source FPS game *QFusion* [14] onto our proxy-server architecture and reporting the scalability results (Sections 5-7).

1. Scalability of Interactive Real-Time Applications

Interactive real-time applications have various requirements regarding scalability. For example, in multi-user training scenarios of catastrophes, physical simulation of fire spread and behaviour simulation of crowds are required to scale. In virtual residence environments, the world size and the storage space for user-created models and textures must scale. This section discusses how scalability of online games is influenced by the real-time game processing and identifies the main dimensions of scalability for games.

1.1. State and its Processing in Online Games

In order for an online game to be interactive and provide the impression of a fluently evolving game world, the game state must be updated in real time: there is a deadline to be met for each state update. A single update incorporates user inputs and the simulation of the virtual world; the update has to be visualized at all participating game clients. The game state consists of the states of all dynamic *game entities* like avatars (virtual representations of human users) and other game elements users can interact with. An

entity's state contains attributes like position, orientation, and the current action, which have to be updated both frequently and consistently: All clients connected to the same session have to present the same game state to their users.

The frequency of the game state updates (so-called *ticks*) defines the *responsiveness* of a distributed online game, i.e., how fast a certain user input is acknowledged, processed and reflected in the game state. The required responsiveness depends on the type of the online game: Fast-paced First Person Shooter (FPS) games require a very high responsiveness due to the direct controlling of the player's avatar in a very tight feedback loop. These games usually run at update rates ranging from 10 to 35 and even more updates per second depending on the actual implementation. In contrast, Role-Playing Games (RPG) are played more indirectly by giving commands to the avatar in a more relaxed feedback loop. Therefore, these games usually run at lower tick-frequencies of 5 to 10 updates per second. However, all these games are still soft real-time systems: A delay of a state update beyond the length of a tick, i.e. a violation of the real-time constraints, immediately destroys the steady flow of the game, leads to stuttering animations, and thus disrupts the users in their game immersion.

1.2. Conventional Client-Server Processing

Currently, the most common solution for small-scale FPS and RTS games is the well-known *client-server architecture*. The single server receives user actions from all participating clients, computes the new game game state for every entity in the game, and sends the new state back to each client. If we abstract the virtual game world to be a two-dimensional area, the single server is responsible for all entities in this complete area as depicted by Figure 1.

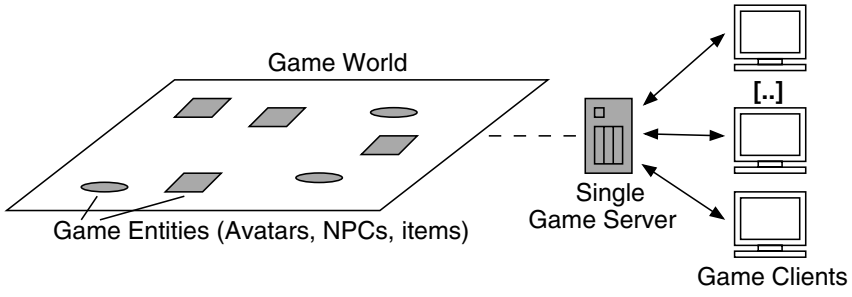


Figure 1. Single-server setup: the server is responsible for all entities in the game world

The single-server concept is easy to implement, but the server constitutes a single point of failure and, more importantly for massively multiplayer gaming, is not able to scale a game session for an increasing amount of entities. If the amount of entities in the game is increased when more clients join, the required computation for a game state update grows: the server will eventually congest and violate the game's real-time constraint. In order to scale a game session for massively multiplayer gaming, the system architecture must be able to increase the computing power, which can be achieved by using several CPUs and/or servers.

1.3. Scalability Dimensions of Online Games

In the following, we identify three main scalability dimensions of Massively Multiplayer Online Games by analyzing different game genres:

1) Overall number of participating users. In MMORPGs, this is the number of users connected to a single virtual world (*realm* or *shard*) which, therefore, are able to interact with each other. In an FPS or RTS, this is the number of users playing in the same game session. Among different game genres, RPGs have the strongest requirement regarding this scalability dimension, because the usually huge game world has to be populated. However, MMO versions of FPS and RTS require this scalability dimension as well, in order to provide game sessions for more than the current maximum of about 100 users.

2) Game world size. This dimension of scalability is especially required by MMORPG, where users can "live" in very large virtual worlds. In order to scale the game world size, two resources are required. The first is processing power for computing actions of actively computer-controlled entities. In a larger game world, more of such entities like Non-Player Characters (NPCs) or mobile enemies (mobs) have to exist and be controlled by the servers. The second resource is the main memory for storing an increasing amount of static terrain geometry and dynamic entities populating the game world.

3) Maximum player density. The density of players refers to the number of players located near to each other in a comparatively small region of the game world. This dimension of scalability is urgently required by FPS and RTS games: Because the main goal of these games is to defeat other human players, users move their players to where some action is going on, and thus dynamically create player clusters with a high density. For MMO versions of FPS and RTS, player density has to be scalable in order to provide responsive gameplay for such situations. In MMORPGs, this dimension may be required as well: For example, users come together at a high density in big cities to interact with each other (trade or chat) or form large armies in high density player-vs-player scenarios.

There are several other scalability dimensions for specific game designs: for example, a virtual housing environment may require scalable disc space for storing user-created and uploaded models and textures. However, the three dimensions 1)-3) identified above are general for all massively multiplayer games and required by the three main real-time game genres of RPG, RTS and FPS.

2. Parallelization Approaches

In this section, we discuss two different multi-server parallelization approaches for online games with regard to the three scalability dimensions introduced in Section 1.3: The *zoning concept* which is already used for MMORPG, and our own alternative *replication concept* suitable for MMO versions of fast-paced FPS and RTS games.

2.1. Game World Zoning

The zoning approach partitions the game world of an MMOG into disjoint zones. The state updates for zones can be computed independently of each other on several servers. The highest performance is reached if each server handles only a single zone: then the architecture can be viewed as an independent collection of single-server topologies.

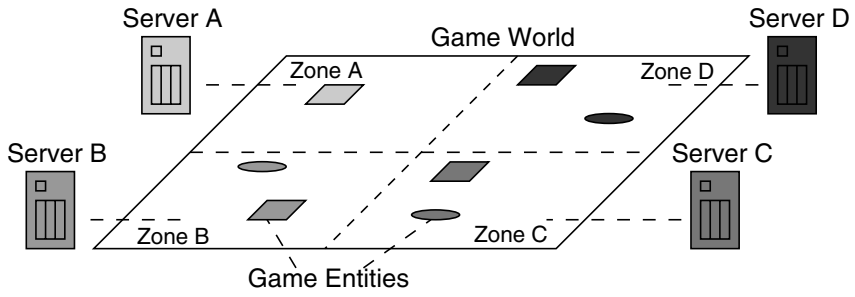


Figure 2. Game World Zoning

Figure 2 depicts an example of a game world partitioned into four zones, where each zone-server handles its entities independently of the rest of the game world. Communication between servers only occurs when a user moves his avatar from one zone into another: The game client has to establish a new connection to the server responsible for the new zone and the servers have to handle the client migration in a consistent way. There are several possible enhancements like dynamic resizing of zones, processing several lightly populated zones on a single server or slightly overlapping zones for a transparent migration of clients. The usual cluster-setup for zoning is discussed in [5], while [7] presents a peer-to-peer-based architecture.

The zoning approach allows to scale the total number of players, i.e., dimension 1), very well if the players are distributed evenly among the different zones. In MMORPG, where zoning is commonly applied, the users are encouraged to spread out in the game world: the zones are usually different in difficulty and obtainable game rewards, such that players of different experience levels tend to become separated. Additionally, zoning scales the game world size (dimension 2) because additional zones and servers can be added arbitrarily. However, zoning does not work well for FPS and RTS games, because it does not scale the density of players (third dimension): "hot spots" with a high density of players congest the single server of the corresponding zone.

2.2. Our Approach: Game World Replication

An alternative to the zoning concept is our approach of game world replication which aims at scaling the user density in interactive real-time applications. In our multi-server replication concept, each server holds a complete copy of the world and all entities of the game, but the responsibility for entities is equally distributed among all participating servers: each server computes the new state only for a particular subset of entities; these are called *active entities* for this server. The entities being updated at other servers are called *shadow entities*. Each server regularly, i.e., after each update, sends the new state of its active entities to all other participating servers which then update their shadow copies of these entities. Figure 3 shows how the replicas at the different servers can be viewed as layers, in which the active entities of a particular server are depicted as filled, while the shadow entities maintained at other servers are outlined.

Only one server is allowed to alter the state of each entity, i.e., has write access to it. This *primary copy* [19] synchronization approach guarantees *eventual consistency* [18] of the replicated game state. In general, there is a trade-off between the responsiveness of the distributed state processing and the degree of consistency. We chose to implement

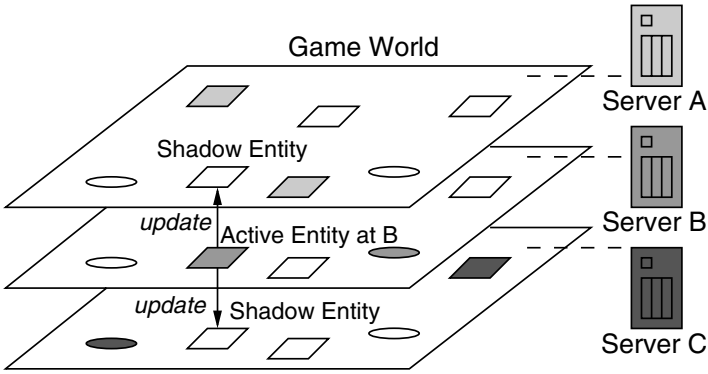


Figure 3. Game World Replication

eventual consistency because it enables high responsiveness of the game state processing (no communication for entity locking is required) at a reasonable degree of consistency.

Our replication approach is particularly suitable for FPS and RTS games because it does not partition the game environment and thus can work on arbitrarily small worlds. As we demonstrate in Section 6, it does scale the maximum density of players (third dimension) since all servers participate in the processing of the entities at a hot spot. The main observation here is that the computation of the new state of an active entity (checking collisions and computing the new position, processing interaction targets, computing health points, etc.) usually takes more time than communicating and applying a remote update of a replicated shadow entity which only requires some unchecked variable assignments. Additionally, the replication approach does not require clients to change their server connection during a game session. Each game server has a full copy of the game state and therefore can serve arbitrary clients regardless of their avatar position in the game environment. This property allows the game to run in a seamless world without zones and therefore is more suitable for small- to mid-sized worlds of FPS and RTS games.

3. The Proxy-Server Architecture

In order to use the replication approach in an actual game implementation, we developed and implemented the *Proxy-Server Network* [10] as an operational multi-server network architecture. In this architecture, several *proxy servers* are interconnected for a single online game session; each proxy can serve arbitrary clients. Servers can be scattered over the Internet at different Internet Service Providers (ISP), and a client chooses to connect to a server with low communication latency: such a server is "close" to the client in terms of latency and thus called proxy in our architecture. Figure 4 shows an example session with four proxy servers.

The proxy-server architecture implements our replication approach (Section 2.2): each proxy holds a complete copy of the application state and is responsible for calculating the new game state of its active entities and for communicating it to all other participating proxies.

The decision about which game entities are active on a proxy determines how well the computational load is balanced, and is based on distinguishing two kinds of entities:

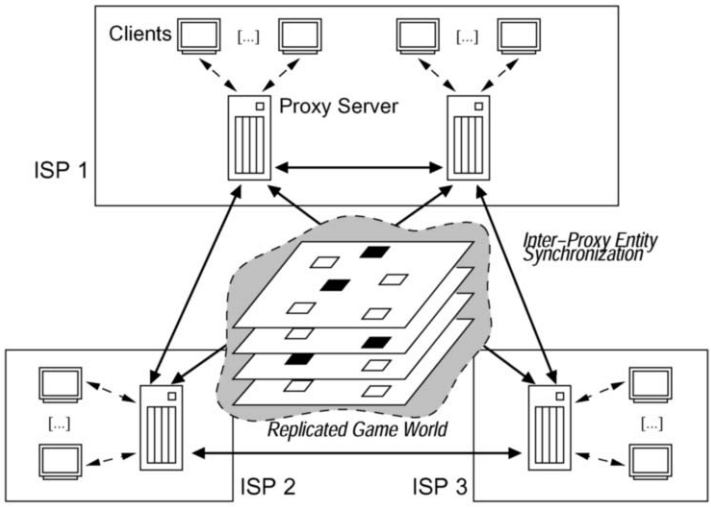


Figure 4. The Proxy-Server Architecture

1. *User-dependent entities* are directly associated to a particular client. These entities are usually the avatars controlled by the user at a client, as well as status and game score information corresponding to that client.
2. *User-independent entities* are not directly associated with a particular client: they are objects in the game world users can interact with, without directly controlling them. Examples are non-player characters (NPCs), or items like health potions, ammunition packs, etc. which can be picked up by users.

The application should distribute the active state of user-independent entities among all participating proxy servers in a way that takes the actual processing power of the server machines into account. The user-dependent entities have to be active entities at the proxy server the particular game client is connected to. This way, all user actions which manipulate the user's own entities (like moving its avatars) can immediately be processed at the client's proxy, resulting in high responsiveness for these frequently issued user actions. The proxy server can immediately change the entity state (e.g., change the position) and confirm the new state to the client. Additionally, the server sends an update message to all other proxies which update their corresponding local shadow entity of the changed entity accordingly.

The processing of interactions can not be performed solely by the proxy server of a particular client: an interaction issued by a user (firing a weapon, picking up an item, etc.) possibly affects not only the state of the user's own avatar entity, but also other entities. In the general case, the interaction target will be active at another proxy, which requires sending the interaction to this remote proxy for evaluation.

Fig. 5 depicts the processing of an interaction example in the proxy-server architecture. The user at client A issues an interaction affecting the avatar of client B, for example, the user fires a weapon onto the position of the avatar of B. In step ①, client A submits the action to its proxy server which validates the received input in step ②. If the validation was successful, i.e., the state of the avatar allows to perform the attack, then the proxy sends an acknowledgement back to client A (step ③) and forwards the interaction to all other participating servers in step ④. The other proxies update their local game

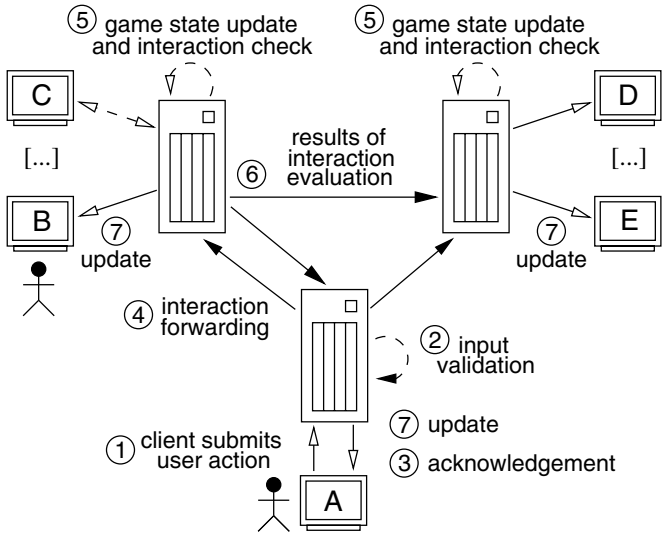


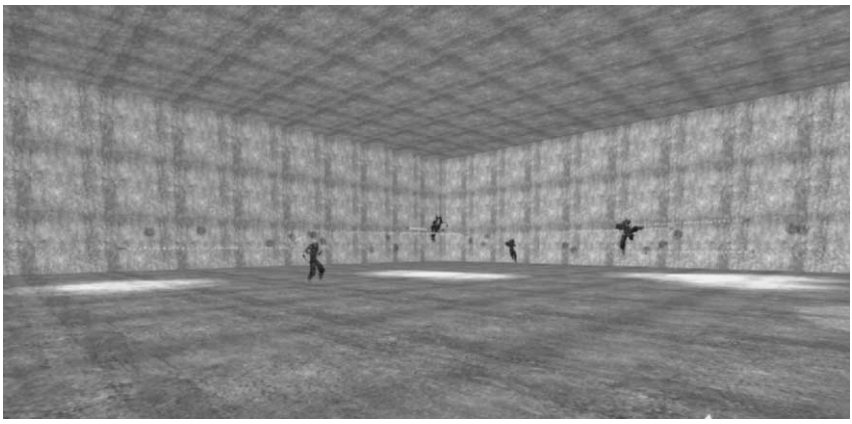
Figure 5. Interaction Processing

state, i.e., they update the avatar's state of the attacking client A in step ⑤. Additionally, each remote proxy checks whether a game element it is responsible for is affected by the attack of avatar A. Let us assume that the avatar of client B is hit by the attack. Then the local proxy of client B updates the state of its avatar by decrementing health points and informs all other proxies about this state change (step ⑥). Finally, in step ⑦, all proxies inform local clients which are directly affected by the interaction (clients A and B). Additionally, all clients whose avatar is located near to the interacting avatars, are notified about the interaction. For example, the users at clients D and E observe the interaction and the proxies inform the clients about it.

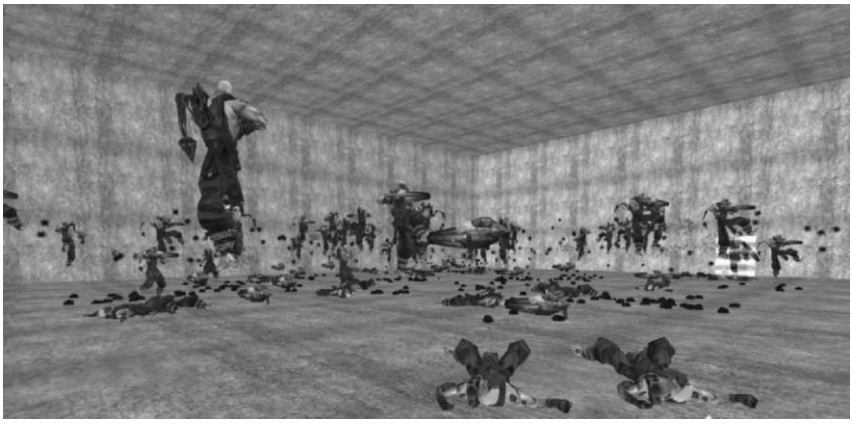
The discussed distributed computation scheme of interactions in the proxy-server architecture might in principle lead to situations where competing user inputs are processed in the incorrect order. In [10] and [12], we discussed in detail the correctness issues and solutions by incorporating additional mechanisms – *local lag* or *timewarp* [8,20,3] – in the proxy-server architecture.

4. The QFusion Engine

For testing the practical impact of our replication approach, we ported the open source FPS game *QFusion* [14] onto our proxy-server topology and studied the scalability of the density and maximum number of participating players. Before discussing the details of the implementation in Section 5, we describe the original game processing. The QFusion-Engine is an enhanced version of the popular FPS game *Quake 2* [17] of *ID Software*. The source code of Quake 2 (implemented in C) has been made open source by ID Software in 2001 and was then used as the basis for the QFusion-engine. Fig. 6 shows two screenshots of five players with the original implementation and fifty players already using the proxy-server port, allowing a high avatar density.



(a) Small-scale session screenshot with 5 players



(b) Large-scale session screenshot with 50 players in a high density

Figure 6. Game Screenshot of QFusion

While QFusion provides much more sophisticated graphics than the original Quake 2, the basic network code remained the same. Therefore, just as Quake 2, QFusion is a very fast-paced, small-scale multiplayer First Person Shooter game using a single server. Each client sends the user commands to the server which frequently (10 updates per second) computes a new game state and sends it back to the clients. This original setup is depicted in Figure 7.

While the single-server architecture works well for small sessions, there is a particular maximum number of players depending on the processing power of the server machine: the server will not be able to maintain the required rate of 10 updates per second if the number of clients exceeds this maximum. Our experiments, which will be discussed in Section 6, showed that the QFusion-server running on a Pentium 4 1.7 GHz host can maintain the required update rate for up to 38 participating clients; more clients result in server congestion: the game becomes unresponsive, movements of avatars and user interactions are constantly lagged.

In order to port the QFusion-engine to our proxy-server architecture, we thoroughly analyzed the game state processing inside the server update loop. The game state is called

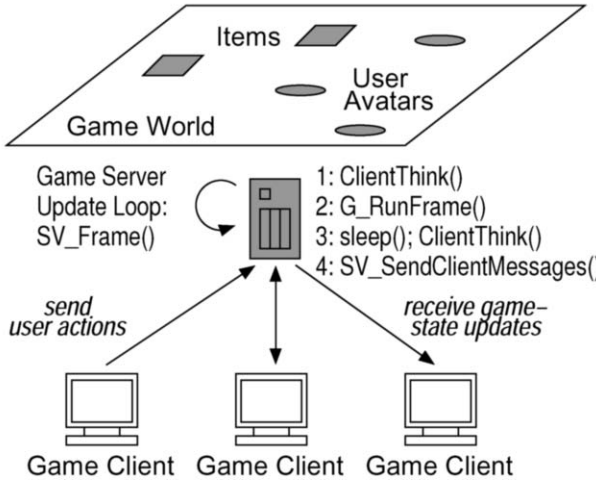


Figure 7. Original QFusion Architecture

`frame` in QFusion. There are four blocks of computation in the main update function `SV_Frame()` which are processed during each update loop, see Figure 7:

1. Receiving, validating and processing user actions in the function `ClientThink()`. The server receives user actions in form of `usercmd_t` structs from the clients. For each of these messages, the server computes the new state of the corresponding user, which is stored as a `player_state_t` struct containing variables for all relevant values like position, movement velocity, viewing direction, the weapon currently used etc. The update of these variables depends on the particular user actions and possible interactions with other users (being hit by a weapon, e.g.).

2. Updating user-independent entities in the function `G_RunFrame()`. All server-controlled items – weapons, armor, ammunition packs etc. – are processed in this step. The server calculates a new state for each of these entities after the freshly passed time of 100 ms which corresponds to the standard tickrate of 10 ticks per second.

3. Waiting for the frame time of 100 ms to complete if there is time left after the steps 1 – 2. During waiting, the server still listens on incoming user actions from clients: if a user action is received, then the server will awaken and will validate and process this action by running the player update function `ClientThink()` for the particular client.

4. Transmission of the new game state to clients in the function `SV_SendClientMessages()`. At the end of the game frame, all the new player and server-entity states computed in the previous three steps are sent to the clients. Instead of sending the complete state to all clients, the server determines and sends the relevant subset of the overall game state to each client. This filtering of information not only reduces the used network bandwidth, but also prevents the type of cheating in which hacked clients can make hidden entities "visible through walls", because the server does not send recent data concerning these hidden entities.

In our port of the QFusion-engine onto the proxy-server architecture, this main loop for the computation of a frame was enriched to incorporate the active/shadow concept. Each server receives and processes updates of shadow entities and generates/sends appropriate update messages for its own active entities. We discuss the required changes to the server processing in the next section in detail.

We only present the server update loop of QFusion and do not discuss the client-side processing: an important feature of our proxy-server architecture is that a client does not need any additional functionality, compared to the single-server approach. Therefore, only adjustments to the original QFusion-server were necessary to incorporate the replication concept.

5. Implementing the QFusion-Engine on the Proxy-Server Architecture

We ported the QFusion-engine onto the proxy-server architecture in order to test our replication approach using several servers, described in Section 2.2, for scaling the maximum number and density of players in fast-paced FPS games. Additionally, it is a demanding case study on how the eventual consistency synchronization affects the game's responsiveness and whether a very fast-paced shooter game is not delayed by the inter-proxy synchronization communication. In the remainder of the paper, the new ported engine will be called *QFusion Proxy-Architecture (QPA)*.

5.1. Incorporating the Active/Shadow Concept

The two types of *user-dependent* and *user-independent entities* discussed in Section 3 can be found in the QFusion-engine as well. Our general solution for the distribution of the active role of entities is that: 1) each proxy is responsible for the user-dependent entities of its locally connected clients, and 2) the active role of user-independent entities is distributed among proxies in a load-balancing way.

For QPA, the active role distribution of user-dependent entities implies that the state of a client can only be changed by the proxy the client is directly connected to. This distribution ensures the eventual consistency of entities as explained in Section 2.2.

5.1.1. Optimized User-independent Entity Processing

The user-independent entities are active on every proxy server in the QPA. Each proxy can alter the state of these entities, and has to send an appropriate update message to other servers if the state is changed. Although the apparently concurrent access to user-independent entities is not synchronized by the architecture itself, consistency is still ensured because the game logic prevents a concurrent access as follows. All user-independent entities in QFusion are passive items (weapons, ammunition, health packs or armor) which can be picked up if a user moves his avatar directly over the item. Due to the usually in FPS games implemented collision handling being also used in QFusion, only one avatar at a time can walk over an item to pick it up.

We implement the processing of an item pick-up action in QPA as follows. The responsible proxy of the avatar that moved over an item removes the item from the map, updates the avatar state (for example, increasing health points in case of a health pack being picked up) and sends an update notification concerning the removed item and the updated avatar state to the other servers. If another avatar maintained at a different proxy tries to walk over the same item at the same time, this proxy will detect the collision with the avatar already "standing on" the item and will not execute that movement of its player. This optimized synchronization using the existing collision detection reduces network communication. It also results in highly responsive execution, because the local proxy performing the movement can immediately perform the pick-up action as well.

5.1.2. Inter-Avatar Collision Handling

Determining and handling a collision of two avatars maintained at different proxy-servers is problematic: depending on the communication latency between the servers, positions of remote avatars might be outdated, yet have to be used for collision-checking of locally active avatars.

In our demonstrator RTS game *Rokkatan* [11], we used a pessimistic approach to solve this problem by virtually extending the collision boundaries of avatars. Collision checks are performed using these bigger boundaries which serve as a "bumper area" avoiding that the actual avatars move into each other due to inter-server latency.

For QPA, an optimistic approach can be applied because users try to avoid colliding each other in an FPS game. Collision checks are performed by each proxy using the original small bounding boxes of the avatars. This might lead to avatars maintained at different proxies moving into each other. As soon as the proxy-servers recognize such a situation by receiving the next entity update message of the other avatar, both avatars are only allowed to move in the opposite direction to solve the collision conflict. This optimistic mechanism turned out to be very suitable for the game style of QPA: users tend to preserve distance to others for being able to move freely, such that inter-avatar collisions are rare.

5.2. Main Loop of QPA

We enriched the original server update function `SV_Frame()` to incorporate the active/shadow concept and to handle synchronization messages with other servers to ensure the eventual consistency of the replicated game state.

Figure 8 illustrates an example setup of the QFusion Proxy-Architecture, the communication between the participating processes and the processing in the enriched `SV_Frame()` function. Each server participating in a single session has a full copy of the game state. During the computation of a new game state, each server has to process its local active entities and to update the remote shadow entities; the resulting processing steps for the different types of entities are printed in bold font in the figure.

In the following, we discuss the main steps of the QFusion-Proxy-Server update loop, referring to Figure 8:

1. Receiving, validating and processing of user actions from local clients. Just as in the original, single-server version, each proxy server receives user actions from the directly connected clients and computes the new state of the corresponding active player entities using the player update function `ClientThink()`. In addition to the single-server implementation, the proxy server will immediately forward the new `player_state_t` structs to all other participating proxy servers if the action of that particular user is an interaction. The other proxy servers need this information about the interaction in order to check whether one of their local clients is affected. This forwarding of interactions is performed in the new function `QPA_SV_SendRemoteCLUpdate()`.

2. Receiving and processing updates of shadow entities from other proxy servers. If the state of a remotely maintained avatar has been updated by the responsible proxy, the corresponding update message including the new `player_state_t` struct has to be processed. In this step, the proxy computes all updates in the function `QPA_Update-RemoteClientState()` by assigning the received updated values of position, health points etc. to its local shadow data structure of that avatar entity. Additionally, the proxy

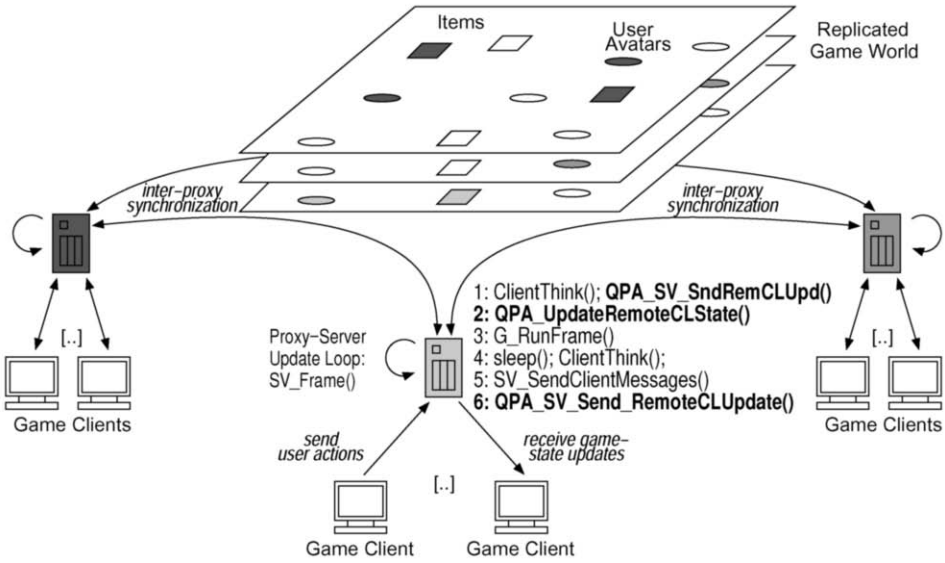


Figure 8. Ported QFusion-Proxy Architecture

checks if a received interaction like the firing of a weapon affects a local active player entity. If there was a successful hit, for example, the proxy will update its local active player entity (decreasing health points, etc.) and send a corresponding update to the other proxies. This way, we implement the general interaction processing scheme of the proxy-server architecture discussed in Section 3 (Figure 5).

3. Updating the user-independent entities of the game world (pickup-items like weapons and armor). As discussed in the previous subsection, each proxy server checks whether such an item is picked up by one of its clients in the function `G_RunFrame()`.

4. Waiting for the complete time of a single server frame (i.e., 100 ms) to pass. As in the original QFusion-engine, the server sleeps the remaining time of the frame and processes incoming user actions during that time by running the `ClientThink()` function.

5. Transmitting the new game state to clients. This part has not been changed in comparison to the original engine: each proxy server determines what new state has to be sent to which directly connected client and transmits this information in the function `SV_SendClientMessages()`.

6. Final synchronization transmission of all changes of active entities which have not been sent in the current frame yet. While interactions already have been forwarded in the first step, non-interactive user actions like movements, changes of viewing directions or changes of the activated weapon have not been sent yet. These entity updates are sent in this final step using the function `QPA_SV_SendRemoteCLUpdate()` to be processed by other servers as soon as they run step 2 of their frame computation next time.

This main loop of our QFusion Proxy-Architecture port incorporates the general concept of the replication approach. The two most compute-intensive functions – the state computations of all active player entities (`ClientThink()`) in step 1 and the filtering of the relevant state information (`SV_SendClientMessages()`) in step 5, – are distributed among all participating proxy servers. However, we introduced some addi-

tional computation for sending and processing entity synchronization messages (`QPA_UpdateRemoteClientState()` and `QPA_SV_SendRemoteCLUpdate()`).

5.3. Communication in QPA

The original QFusion-engine uses its own network subsystem, in which clients and the server directly communicate over the sockets API as illustrated in Figure 9.

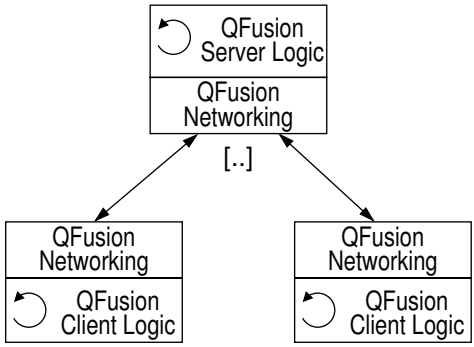


Figure 9. Original QFusion Networking

In order to synchronize the replicated game state of a QPA session, additional inter-server communication had to be incorporated into the QFusion network layer. Instead of adding this functionality to the original network system, we replaced the complete communication layer of the QFusion-engine with our *Game Proxy-Architecture (GPA)* communication library as illustrated in Figure 10.

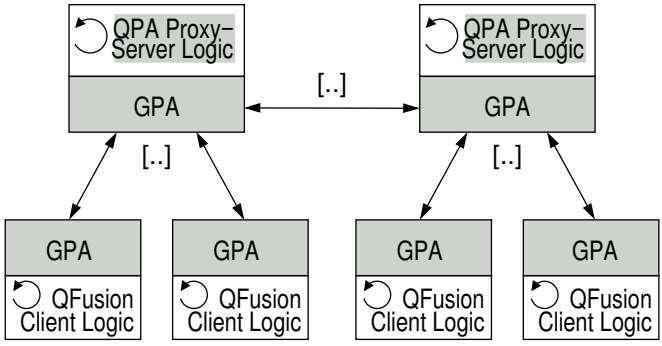


Figure 10. Ported QFusion using the GPA

The GPA library has been implemented in our previous work using C++ to ease the use of the proxy-server architecture by providing socket-based channels for client-server and inter-server communication. For each communication channel, the processes can send and receive character-arrays using an API with several degrees of communication reliability. The inter-server communication uses IP-Multicast and automatically falls back to unicast communication for servers which are not able to join the multicast channel because some intermediate router does not support multicasting. Figure 10 shows the

altered server main loop and the replaced network subsystem shadowed, while the client logic of QFusion was not changed.

The GPA was already successfully used as the network subsystem in our RTS demonstration game Rokkatan [11] and allowed us to add the required additional communication functionality to the QFusion-engine with relatively little work.

6. Experiments

We conducted different tests of the presented QFusion Proxy-Architecture in order to verify its functionality and, especially, to evaluate its scalability, i.e., the actual increase of possible player numbers in a single session.

6.1. Functionality and Responsiveness Test

For testing functionality, we set up a distributed session with two QFusion proxy-servers located in Muenster and Luebeck at the distance of ca. 500 km, see Figure 11.

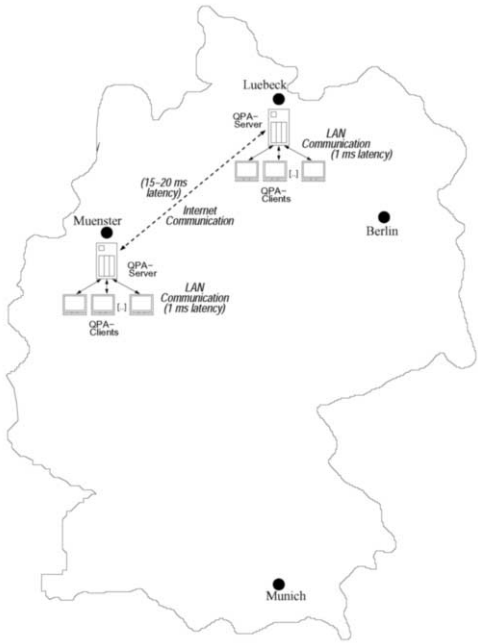
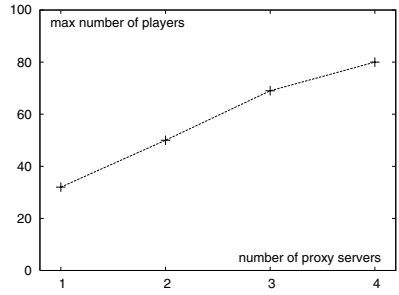
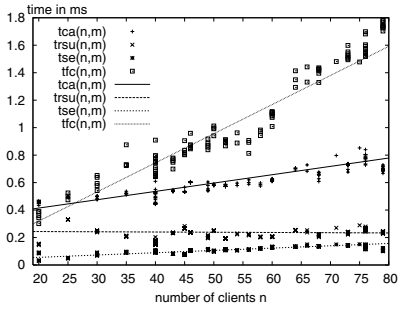


Figure 11. Functionality Test Setup

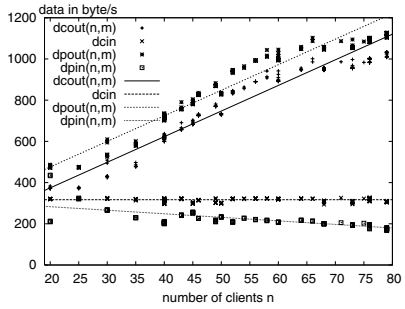
At each of the two sites, several clients are connected to the server near to them in terms of communication latency. The proxy-servers fully replicated the game state and synchronized their entities according to our active/shadow approach over a public Internet connection with an inter-proxy latency of about 15-20 ms. The distributed play test ran successfully, there was no noticeable delay of interactions between users at different sites. Avatar movements were processed in a highly responsive manner due to the low communication latency of clients to the local proxies.



(a) Max. Players for Increasing Number of Servers



(b) Computation Times, Four Servers



(c) Bandwidth Use, Four Servers

Figure 12. Results of the various Scalability Experiments

6.2. Scalability Tests

For testing the maximum number of simultaneous players, we used an autonomous non-graphical bot client of which we started several instances on each single client host. The bot client generates as many actions as possible by continuously moving around and firing weapons as fast as possible, i.e. we tested the worst case: at each tick actions of all users are processed. Due to the high number of required hosts for the clients, coordinating and running the tests was quite complex and took dozens of hours. For easier management, we ran the scalability test for different numbers of proxies in a single LAN environment in Muenster. Using up to 20 hosts at our department for servers and clients, we were able to test sessions with up to four participating proxies. The server machines in the tests were single-CPU Pentium 4, 1.7 GHz hosts. For each setup, we continuously added clients to the session until the servers became congested, i.e., were not able anymore to finish the complete tick calculation in the time of 100 ms given by the QFusion standard tickrate.

The maximum number of players before the servers saturated is illustrated in Figure 12(a) demonstrating a scalable behaviour for up to 80 players on four servers. In numerous test runs, we did not only measure the overall number of clients, but also obtained fine-grained timing and bandwidth use values, shown in Figures 12(b) and 12(c). For an increasing number of clients, the lines in Fig. 12(b) and 12(c) show the computation time of four main steps of the loop described in Section 5.2 and bandwidth use to other processes at a single server. The measured values allow a detailed analysis of the factors influencing the QPA scalability; a corresponding discussion can be found in [9].

The tests were run on a very small quadratic map without any additional separating walls, because we wanted to test the actual scalability of player density. In the single-server setup of QPA, up to 32 users were able to participate in an uncongested session. Using four servers, the parallelized QPA version allows to almost triple the maximum number of participating users to up to 80 players. In comparison, the original, highly optimized QFusion-engine on the same server host is able to support 38 users, i.e. the overhead of the additional proxy-related processing is about 19 %.

7. Related Work and Conclusions

In the area of online games as an important application in the area of virtual environments, different dimensions of scalability required by particular game types are critically important. The commonly used zoning approach scales the overall size of the virtual environment as well as the total number of users, as long as they are distributed over the independent regions equally. However, the zoning concept does not scale the density of users in the virtual world, which disqualifies this approach for FPS and RTS games. The replication concept presented in this paper provides a strong improvement regarding the scalability of player density and allows to substantially increase the amount of concurrent users in game applications with small virtual environments not suitable for zoning.

The presented port of the QFusion-engine demonstrates that our replication approach on top of the proxy-server architecture is a feasible method to scale the overall number of players as well as their density. While the zoning approach as discussed in [5,7] targets MMORPGs and is favourable for scaling the total number of players in large game environments (our identified scalability dimensions 1 and 2), it is not suitable to scale the density of players which is critically important for fast-paced action games. The total player numbers we achieved with QPA are still much lower than those of available MMORPGs. We expect that MMO-gaming in FPS games will involve hundreds rather than thousands of users, due to their much smaller game world.

The Colyseus system [4] also uses replication, but entities are replicated on demand, which requires an additional setup time and targets large game environments. Regarding our presented scalability dimensions, [16,15,6] provide some related, but less detailed discussion. [2] presents a detailed scalability analysis for the game *Quake 1* as the predecessor of Quake 2 used in our implementation.

Our goal was to scale the density of players, which we achieved by processing an increased number of simultaneous avatars in a very small test environment using several servers. Additionally, setting up local proxy-servers allowed to run the game at a high responsiveness. An alternative way of increasing the number of players is to use a server of higher performance, i.e. an SMP system as discussed in [1] for Quake 1, using a pessimistic locking mechanism for synchronizing concurrent write access. Both possibilities – SMP servers and multiple servers – are orthogonal to each other and can be used either alternatively or in combination.

The sequential implementation of the commercially successful game QFusion is highly optimized, such that porting the server onto the proxy-network introduced an overhead of about 19 % for the additional processing. As we experienced while implementing our demonstrator game *Rokkatan* [11], the implementation can be more efficient when incorporating the replication approach from scratch, because the data struc-

tures can be designed and optimized accordingly. Despite these difficulties, the QPA port does scale the entity density quite well for a reasonable cost. The replication approach is currently being incorporated in the *Real-Time Framework* [13] middleware of the EU IST *edutain@grid* project[1], where replication and zoning will be combined for achieving dynamic scalability of real-time interactive application in a grid-computing environment.

References

[1] Ahmed Abdelkhalek and Angelos Bilas. Parallelization and performance of interactive multiplayer game servers. In *IPDPS 2004*, Santa Fe, New Mexico, USA, April 2004. IEEE.

[2] Ahmed Abdelkhalek, Angelos Bilas, and Andreas Moshovos. Behavior and performance of interactive multi-player game servers. In *Proceedings of 2001 IEEE International Symposium on Performance Analysis of Systems and Software*, Tucson, Arizona, USA, November 2001.

[3] Yahn W. Bernier. Latency compensating methods in client/ server in-game protocol design and optimization. In *Proceedings of Game Developers Conference'01*, 2001.

[4] Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. A distributed architecture for multiplayer games. In *PACM/ USENIX NSDI*, San Jose, USA, 2006.

[5] Wentong Cai, Percival Xavier, Stephen J. Turner, and Bu-Sung Lee. A scalable architecture for supporting interactive games on the Internet. In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation*, pages 60–67, Washington, D.C., May 2002. IEEE.

[6] Thomas A. Funkhouser. Network topologies for scalable multi-user virtual environments. In *VRAIS '96: Proc. of the Virtual Reality Annual Int. Symposium*, Washington, DC, USA, 1996. IEEE.

[7] Björn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-peer support for massively multiplayer games. In *IEEE Infocom 2004*, Hong Kong, China, March 2004. IEEE.

[8] Martin Mauve, Jürgen Vogel, Volker Hilt, and Wolfgang Effelsberg. Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia*, 6(1):47–57, February 2004.

[9] Jens Müller-Iden. *Replication-based Scalable Parallelisation of Virtual Environments*. PhD thesis, Universität Münster, July 2007. <http://pvs.uni-muenster.de/jmueller/mueller07.html>.

[10] Jens Müller, Stefan Fischer, Sergei Gorlatch, and Martin Mauve. A proxy server-network for real-time computer games. In *Euro-Par 2004 Parallel Processing*, volume 3149 of *LNCS*, pages 606–613, Pisa, Italy, August 2004. Springer.

[11] Jens Müller and Sergei Gorlatch. Rokkatan: scaling an RTS game design to the massively multiplayer realm. *ACM Computers in Entertainment*, 4(3):11, 2006.

[12] Jens Müller, Andreas Gössling, and Sergei Gorlatch. On correctness of scalable multi-server state replication in online games. In *NetGames 2006*, Singapore, October 2006. ACM.

[13] Alexander Ploss, Frank Glinka, Sergei Gorlatch, and Jens Müller-Iden. Towards a high-level design approach for multi-server online games. In *GAMEON'2007*, Bologna, Italy, November 2007.

[14] Open Source Project. Qfusion engine <http://sourceforge.net/projects/l33t/>.

[15] Jouni Smed and Harri Hakonen. *Algorithms and Networking for Computer Games*. Wiley & Sons, 2006.

[16] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. Aspects of networking in multiplayer computer games. In *Proceedings of International Conference on Application and Development of Computer Games in the 21st Century*, pages 74–81, Hong Kong SAR, China, November 2001.

[17] ID Software. Quake 2 <http://www.idsoftware.com/games/quake/quake2/>.

[18] Andrew Tanenbaum and Marten van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[19] Matthias Wiesmann, Fernando Pedone, André Schiper, Bettina Kemme, and Gustavo Alonso. Understanding replication in databases and distributed systems. In *Proceedings of 20th International Conference on Distributed Computing Systems (ICDCS)*, pages 464–474, 2000.

[20] Hai-Rong Yan, Ya-Chong Zhang, Guo-Ji Sun, and Lian-Jiong Zhong. Research on time warp mechanism in HLA. In *Proc. of the 2nd Int. Conf. on Machine Learning and Cybernetics*, pages 1092–1095, 2003.

[1] www.edutaingrid.eu

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

ImageGrid: An Image Processing Grid Based on CGSP

Hai JIN, Ran ZHENG and Qin ZHANG
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China

Abstract. Grid computing has emerged as a new infrastructure for modern image processing applications with the demands of more large-scale, collaborative processing and storage capabilities. But it is too hard to develop grid applications with the immature grid-enabled computing environment. ImageGrid is a project in ChinaGrid to provide an efficient and collaborative grid processing environment on components and workflow technologies. In this paper, the infrastructure and the integrated mechanisms are discussed, which provide the benefits of flexibility, reusability and scalability. The deployments and realizations of typical image processing applications in ImageGrid are also given out.

Keywords. Grid Computing, Image Processing, Grid Workflow, Digital Virtual Body, Remote Sensing, Medical Information Integration

Introduction

0.1. Motivation: Why use ImageGrid?

Image processing is a very exciting area of computer science research, which utilizes computational and digital analytical methods to solve image problems, such as image enhancement, restoration, compression and analysis. There are many useful applications: remote sensing, medical applications, intelligence gathering/'smart' weapons, car guidance and optical correction, which are growing and non-exhaustive.

Many image processing applications create numerous 2D/3D scenarios or models and require high, even exponential increasing computing powers. For examples, stereo image matching and feature extraction are usually too compute-intensive to be done in reasonable time on a single processor. Rendering photorealistic images and computing animated films consume aggregate power of all workstations. It will take several hours or even days for these image processing applications on desktop computers. Global grid is the way to speed up the large-scale processing requirements, which can harness distributed resources such as servers, workstations, clusters and bulky spaces.

What's more, global grid [1] can eliminate the gulfs between image processing and high-performance computing [2]. Many professional researchers of image processing expect to utilize high-performance computers to resolve image processing applications

and upgrade their researching faculties. But they know little about computer, distributed and parallel computing technologies, so that they must spend more energies and times on professional knowledge of computer and computing before working, which is very difficult for them to master better in a short time. Although more high-performance parallel servers and workstations with hundreds of computing powers are unused, and the distributed and heterogeneous characteristics of them also cumbers the combination of high-performance computing and image processing.

There are comparability and inheritability in the research of image processing. Some basic image processing algorithms can be applied into various different applications. For example, image partitioning is available in remote sensing application for target extraction, which is also used in medical, geography and other areas. Once a high efficient algorithm is designed, it can be treated as one of general methods in a basic toolkit to settle some kindred problems. In this way, the latter can continue to research upon the former shoulder. With the increasing of image processing complexity and diversity, it becomes impossible to contain all image processing methods in image toolkit, so that the toolkit should be scalable and extendable.

0.2. The System: What is ImageGrid?

As the reasons above, grid computing are used effectively to advance image processing research. Grid computing has been developed with the high speed of Internet, which are becoming an attractive and promising platform for solving large-scale computing-intensive scientific problems. In grid environment, large numbers of geographically distributed resources are logically coupled together, sharing all computing resources, storage spaces and data resources. Grid Computing can satisfy image processing demands sufficiently, and enable image processing on disparate data and heterogeneous clusters or machines. It shields the heterogeneous bottom of grid computing technology and operating complexity, users can conveniently make use of diverse resources in grid.

0.2.1. ImageGrid Introduction

Image Processing Grid (ImageGrid) [3] is one of five important application grids supported by ChinaGrid, aiming to create a grid platform for image practitioners and researchers to solve large-scale scientific problems of image processing.

ImageGrid is a service-oriented architecture whose aim is to create a grid platform, bridging the gaps between image processing and high-performance computing. ImageGrid aggregates all computing powers and resources as a virtual computer, solving large-scale scientific problems of image processing. It enables users to take advantage of substantial computing powers available on heterogeneous grid environment without in-depth knowledge of parallel computing theories or experiences. Namely it not only provides high-speed executions of complex applications, but also provides an easy-to-use web GUI, through which any user can access ImageGrid from anywhere at anytime as conveniently as Web. ImageGrid provides the software toolkit with the thinking of "write once, anybody run", which enable any scientist or engineer quickly and easily build a distributed parallel application with the combination of his new service component and reusable software components existed in ImageGrid.

The overview of ImageGrid is shown in Figure 1. ChinaGrid Support Platform (CGSP) is the foundation of the platform and applications. Reconstruction of digital vir-

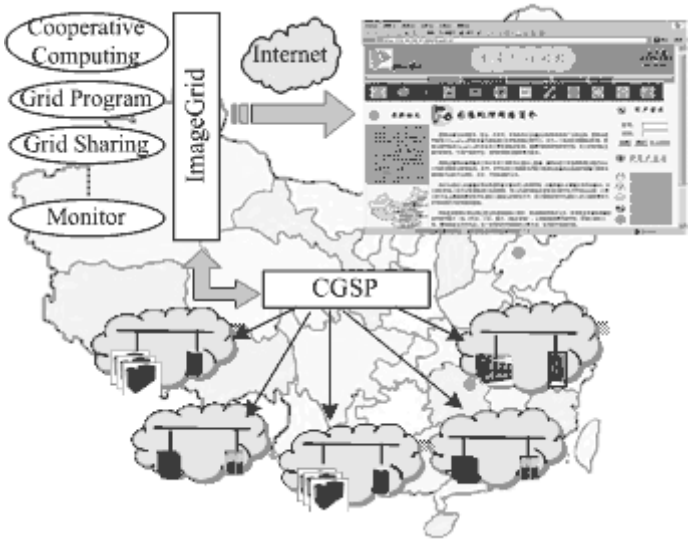


Figure 1. ImageGrid Overview

tual body, image processing of remote sensing, medical image diagnoses are three typical image processing applications in ImageGrid.

0.2.2. CGSP: The Foundation of ImageGrid

ChinaGrid Support Platform (CGSP) [4] is a grid middleware developed for the construction and evolution of ChinaGrid [5]. CGSP aims to integrate all sorts of heterogeneous resources distributed over CERNET, and provide transparent, high performance, reliable, secure and convenient grid services for scientific researchers and engineers. In addition to supply to ChinaGrid, CGSP offers a whole set of tools for developing and deploying various grid applications.

The main characteristics of CGSP are as follows:

1. Based on OGSA (Open Grid Service Architecture) [6], CGSP is developed according to the latest grid specification WSRF (Web Services Resource Framework). It supports multiple service types, such as super (virtual) services, common web services, WSRF services.
2. ChinaGrid has the feature of highly localization. In order to obtain resources nearby and satisfy the autonomy requirement of specialized grid application in ChinaGrid, CGSP brings up the concept of "region" and each region could offer specific services independently.
3. Scalability of CGSP satisfies the demand of expansion of ChinaGrid. The tree structure of CGSP ensures that ChinaGrid has the capacity to cover the majority universities in China.
4. CGSP guarantees the integrity and uniformity of ChinaGrid by a global monitoring system starting from the root region, a uniform entrance, and independent portals in each sub-region.

0.2.3. Typical Image Processing Applications

There are three typical image processing application in ImageGrid: digital virtual body reconstruction, remote-sensing image processing, medical image diagnoses.

The Virtual Human Project is the creation of complete, anatomically detailed, three-dimensional representations of the normal male and female human bodies. The realization of digital virtual human is to identify and classify inner viscera, rebuild the edge profile data of outer body and three-dimensional profile data of inner viscera, and construct high-precision digital human grid to obtain visualization of them. Because the data amount increases greatly in mesh construction with the massive body slices, only grid with high-performance resources can afford the processing.

Geographers use remote sensing technique to monitor or measure phenomena found in Earth's lithosphere, biosphere, hydrosphere, and atmosphere. Remote sensing imagery has many applications in mapping land-use and cover, agriculture, soils mapping, city planning, and other uses. Remotely sensed data are usually digital image data. Therefore data processing in remote sensing is dominantly treated as digital image processing, which includes image preprocessing, advanced processing and categorization.

Medical imaging processes are dedicated to the storage, retrieval, distribution and presentation of images, which are handled from various modalities, such as ultrasonography, radiography, magnetic resonance imaging, computed tomography and so on. Doctor and his patients can get more reliable and precise medical images (such as X-ray slice, CT and MR) with more quick, exact and simple mode, so as to increase the rate of inchoate morbid detection and diagnoses and the living opportunity.

1. ImageGrid Platform

1.1. Holistic ImageGrid Architecture

ImageGrid is built upon campus grids in twelve universities of China as a part of ChinaGrid. ImageGrid screens the heterogeneity of resources, and provides transparent access for grid users to enable image processing grid applications. It makes full use of grid resources of campus grids and interacts with them by standard interfaces. The hierarchical architecture of ImageGrid is shown in Figure 2.

ImageGrid includes three layered parts logically from the bottom up: Grid resources configured with CGSP, ImageGrid application platform and grid portal.

Grid resources, as the lowest layer of ImageGrid, comprise various clusters, workstations, high-performance computers, databases, and large-scale storages in ChinaGrid. They are isolated logically, but can be unified to be shared, integrated and interoperated by CGSP in wide-area grid environment, with which standard interfaces and basal toolkits are offered for upper application platform.

ImageGrid application platform is the most important and pivotal middleware in ImageGrid. It can be divided into two parts with different orientations: ImageGrid core services and ImageGrid fundamental services. ImageGrid core services are the continuation and enhance of CGSP, which are responsible of providing computing powers with the lower interfaces for the share, integration and interoperation. The details of grid resources are hidden as far as possible, and user can access them transparently. ImageGrid fundamental service aims to provide flexible supports to various image-processing appli-

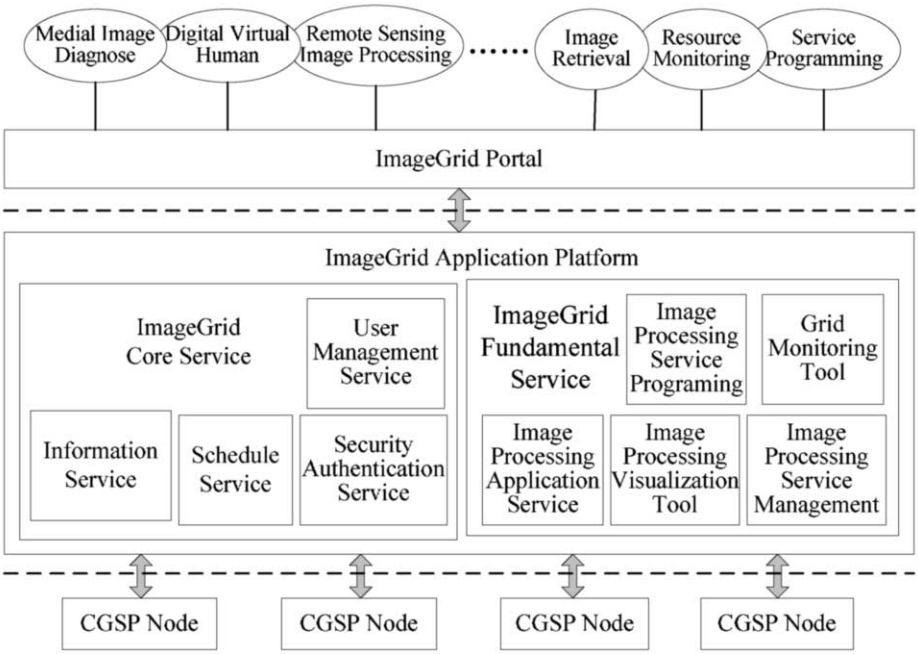


Figure 2. ImageGrid Architecture

cation scenarios by some APIs and SDK such as image processing integrated developing environment, cooperative parallel tasks, application monitoring, so that users can establish their custom-built individual applications or services. Some modular image processing toolkits are also offered for several special requirements. For examples, image visualization tool can present 3D rendering or reconstructing images from remote image services or workstations, image searching can find suitable pictures with the content and character from grid image databases.

ImageGrid portal is the entrance for omni-directional uses, managements and monitor of the platform as a uniform, simple and friendly web access interface. The explicit functions, impressive expression and simple operation can bring convenient use without more knowledge of grid and documents for help. Portal interface is the bridge between ImageGrid application platform and portal, including both ImageGrid core services and fundamental services.

1.2. Technology Features

There are many technologies and methods used in the design of ImageGrid. Here is a generalization of the features, and the details will be expatiated in next section.

ImageGrid shields the distributed locations and heterogeneous characteristics of grid resources with grid platform, screens the complicated inner technical and design details with application middleware, and offers friendly graphic interface and convenient environment with web portal. All of these decrease the difficulty for image processing professional researchers to utilize the platform and toolkits.

ImageGrid provides a series of toolkits and basic services for image processing applications. Computing powers are integrated by grid service components to solving large-

scale scientific problems, yet the difficulty and complexity are reduced by image processing solving environment and workflow engine. Image processing solving environment is based on grid service technology, in which image processing toolkits and software are deployed as service components. All these services are usually deployed redundantly for fault-tolerance, availability, performance and so on. Users just only care for these virtual services of image applications, which are the abstract of several physical services and transferred into different physical services running in our grid. Meanwhile, grid workflow engine can disperse and simplify application logic and cut down the user intervention and operation, so as to improve service performance, reusability and expansibility and even support distributed applications at the same time.

Service Request Broker, a communication tool for grid service components and grid workflow engine, provides a general service transferring interface for remote client. It can replace normal grid client programs, so that grid programmers should do nothing for service caller. It simplifies service calling mode, and enhances the stability and security of grid service components and grid programs.

ImageGrid provides a multi-node cooperative mechanism for image processing applications in grid environment. High-performance computing powers in wide areas, such as parallel clusters, workstations or mainframe computers can be utilized together for various applications, which can heighten both the work efficiency for cooperative image processing applications and the availability ratio of idle resources.

ImageGrid provides global resource views at application level. It can monitor application services, service components and dynamic workflow processing effectively, with which not only users can conveniently query and monitor the whole environment or related grid information at any time, but also the administrator can hunt and adjust more appropriate parameters or better means for image processing applications.

ImageGrid also provides intuitionistic, photorealistic and interactive remote visualization tool. A classical image based rendering (IBR) technology is employed, which is called panoramic cylindrical mosaics for planar images without any depth information. This method can bypass the difficulty of modeling and rendering process of traditional walkthrough system and don't affect the photorealism of scene. Users can utilize this tool to look over the remote image results with huge data and get real-time transform for the display of the result, which not only saves user's storage space, but also ensures the data secrecy and the alternation between user and system. This tool can also be applied to other similar applications such as virtual museum, virtual park.

2. Image Processing Solving Environment

The aim of image processing solving environment in ImageGrid is providing a friendly platform to solve image processing problems easily. Grid service workflow model is commonly used in the research of service-based grid application with the advantage of cross-platform and flexible expression, but the weaknesses are still existent as follows:

1. The transparency is not enough for grid users. While editing the applied workflow, users have to select grid services explicitly and build exact process, which is at variance with the original intention of grid-transparently utilizing grid resources.

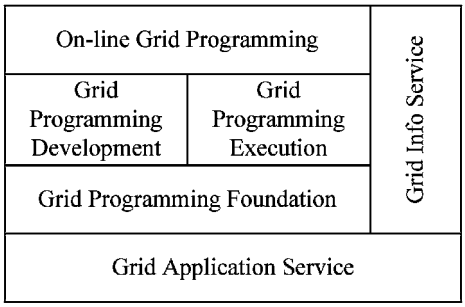


Figure 3. Hierarchical Structure of Image Processing Solving Environment

2. The executing fashion is not flexible for grid user, which is the immediate result of the first weakness. Grid services and accessing resources have been specified by application workflow [7], so that load balance, performance optimization and fault tolerant can not be achieved without resource scheduling during the execution of the workflow.

Conquer these weaknesses and combine the characteristics of image processing applications, service-based image processing solving environment makes image processing grid application developments convenient utilization, fault tolerance and automatic optimization. Grid components and workflow technologies are used to construct and manage the simple/complex applications with various requirements.

2.1. Hierarchical Structure

Image processing solving environment [8] targets image analysis and visualization in distributed grid environment, which is provided for deploying existing image software or toolkits as service components, and building composite image applications by "drag-and-drop" interfaces. Some deployed image processing grid services have been integrated beforehand in ImageGrid. With the recombination of these service components, all simple or complex image processing applications can be processed if decomposed services are supported in grid environment.

Every kind of components, an aggregation of grid services with the same functions and interfaces, will be encapsulated as "virtual service". Each component is deployed in different grid resources with its own service handle, called "physical service". Virtual service masks the details of the lower implementation and simplifies the difficulty of the usage, while physical service is employed when it is called by service request broker.

The hierarchical structure of service-based image processing solving environment is shown in Figure 3, which includes four layers from bottom up.

Grid Application Service Layer is the resource foundation of the platform. It manifests as grid service and its description, which is an encapsulation of software and toolkits in image processing application field.

Grid Programming Foundation Layer is the abstract of grid application service. It makes service implementation transparent to grid users, and supplies grid programming interfaces for image processing. It hides the diversity, physical redundancy and dynamic action of application services, supports parameters transferring and uniform service calling from abstract interfaces to physical grid services. Grid Programming Interfacing

Layer includes Grid Service Management, Grid Service Scheduling, Grid Service Request Broker and Grid Data Management modules.

Grid Programming Development and Execution Layer is the core layer. Grid Programming Development can manage grid application programming interfaces and grid programs with these two sub-systems. When one request is submitted, Grid Programming Execution explains and parses with grid workflow engine, and employs lower interfaces to perform in grid resources, which contains workflow modeling translation and workflow execution engine.

On-line Grid Programming Layer supports grid access for users. A friendly web-based interface is provided for grid users to create their own workflow applications with graphic environment, query grid or workflow states information, even amend their tasks.

Grid Info Service is run through the platform, with which the necessary information of resources, services, tasks or processing procedures can be obtained by users, grid scheduling or other kernel modules.

2.2. Grid Programming Foundation

2.2.1. Grid Service Management

Grid service management module is responsible for the register and repeal of grid services of image processing afforded by different research organizations and schools. Moreover, the statuses or information of registered services should be monitored for resource providers and grid users. Besides, grid manager should pause or delete the lawless grid services which do not obey the service providing protocols.

2.2.2. Grid Service Scheduling

Sometimes several service providers offer grid services with the same functions and interfaces. Once one user wants to utilize them, one should be selected. Grid service scheduling module chooses the right service according to the status of grid resources, the qualities/costs of services or users' requests about the performance of the service. In this module, four scheduling algorithms are supported for choosing as follows:

1. RANDOM : Random Scheduling Strategy.
2. ROUND-ROBIN : Round-Robin Scheduling Strategy.
3. MIN-EXEC-COST : Scheduling Strategy with Minimal Executing Time.
4. MIN-TRAN-COST : Scheduling Strategy with Minimal Transferring Time.

2.2.3. Service Request Broker

There are large numbers of various grid service of image processing in ImageGrid with different interfaces and parameters. They are executed by their own client programs. Generally, client programs are designed differently with different grid services and different interfaces. Once one new service is registered, a new appropriate client program will be supplied. The more the count of grid services is, the more tedious and onerous the work is. A common service request broker is designed for this shortage, which can auto-complete client details with a common interface. It can mask difference of grid services on lower layer, access various servicing handles in grid environment, and simplify service-invoking method. The service request broker facilitates grid users and enhances the stability, feasibility and performance of grid services synchronously.

As a more unattached module in ImageGrid, the common service request broker is independent from other parts and can to be loaded easily. It designed as Client/Service mode, among which the server is the distributed resource as various grid services, and the client is the web portal accessing the lower services.

The processing is as follows: For every service request extracted from users' application requirements, the scheduling module of web portal parses them (virtual service) and transmits to the client program of request broker; the client chooses proper service (physical service) and submits the forming service requests to broker server, namely the server programs on grid resources; the server sends corresponding instance handles of grid services to the client after verifying the validity of the requests. Thus, the client can access the service instance, execute the service and obtain the results.

2.2.4. Grid Data Management

The processing objects of image processing applications are mass data, which may be a single large file or a set of large amounts of small files in distributed storage resources. During the processing of various applications, these data will be transmitted among grid nodes more than once. And the larger the amount of data is, the more aculeate the confliction of naming is, which should be solved as one of the most all-important problems. Grid Data Management Module is to manage image data and provide an efficient and fault-tolerated transmission mechanism.

2.3. Grid Programming Development and Execution

Most image-processing applications are data-intensive. The processing structures of applications are implemented as a set of components that exchange data through stream abstractions. Components are connected via logical workflows, which denote a directional data flows from one component to another.

2.3.1. The Model and Flow Chart

The model and flow chart should support the following functions(shown in Figure 4):

1. Workflow translation from users' submission to workflow language recognised by lower processing programs;
2. Workflow procedure control by workflow modelling and engine during workflow execution;
3. Various workflow management and information services.

The model is composed of four parts: Workflow Translating, Workflow Modeling, Workflow Engine and Workflow Management, which are corresponding to the three above functions.

User requests are submitted from on-line programming graphical portal, but they can not be directly recognized by workflow engine. It must be translated into grid language with EDAG (Extended Directed Acyclic Graph) format, which is the extension of DAG [9], supported in ImageGrid. Only after the conversion and other checking of validity and security, the requests can be processed by other modules.

Workflow Modeling is responsible for constructing grid workflow describing the processing of image application, and decomposes into several activities. The processing

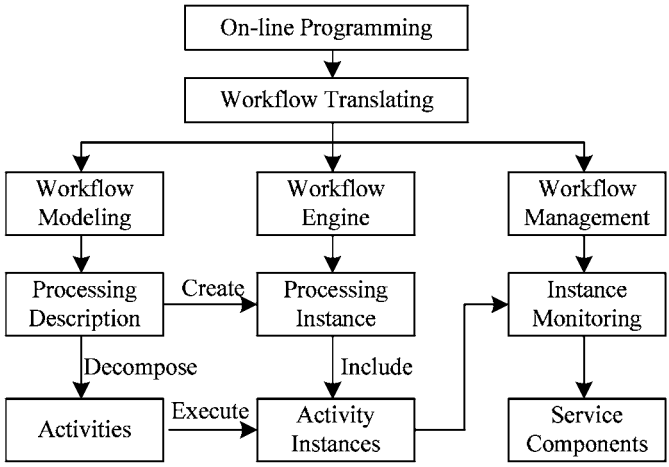


Figure 4. The Model and Flow Chart

is the composition of activities, which are encapsulated as grid service components. The activities are convenient to regroup and model new grid workflows.

Workflow Engine takes care of dynamic execution of processing description after workflow translation, including instances creating, execution managing and parameter transfering. Processing instance is the dynamic creation of processing description. Each instance represents separately as a processing or an activity.

Grid applications are all complex with long periods and protean, uncertain executions. Any more, plenty of applications are maybe submitted to the platform contemporaneously. Workflow Management should manage these numerous application tasks and afford some information services. The main functions includes: 1.Implementing the management of grid tasks and providing some operations for grid users, such as deleting, repairing, and canceling; 2.Monitoring and controlling workflow executing procedure; 3.Collecting service information real-timely for workflow modeling and engine; 4.Supporting fault tolerance.

2.3.2. Data Flow and Service Types

In the workflow model, data resources are exchanged among workflow components. One component requires reading data from its input streams (the output of previous component) and write data to its output streams (the input of next component). Figure 5 illustrates a scenario of a logical workflow with 3 components. In the case, an image application is complex that needs a workflow combining with 3 components (maybe they are not executing in one grid node). Suitable service components are selected by grid flow controlling engine to compose a linear workflow and data stream is flowing among these components.

Here workflow engine affronts two service types: Virtual Service and Physical Service. Users only see virtual services with their functions and parameters regardless of where or how they are implemented. They basic attributes of virtual services are:

1. Virtual service is an independent unit, whose design, deployment and configuration are realized interiorly;

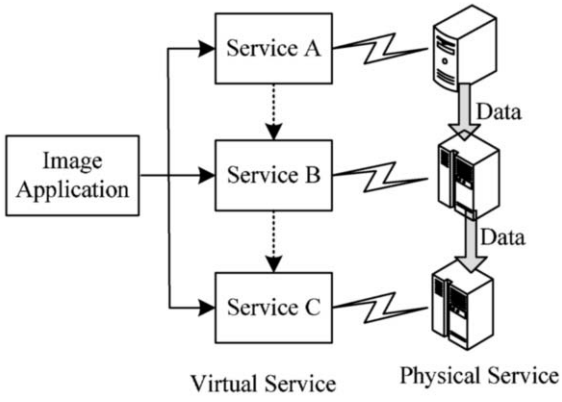


Figure 5. Logical Data Flow and Service Types

2. Virtual Service is separated from outer environment and other components. It means the services are encapsulated strictly, and it is not necessary for users to know the inner details, which are independently of out interface and invoking mechanism;
3. Virtual Service can be reused and assembled to several different applications. Although the kernel of component management is small, the components set (or services set) can be extensible and scalable with the customization of components.

After Grid user chooses what he/she needs from virtual service sets, all the other things are disposed by workflow engine, such as service checking and service matching. Service matching is the mapping from virtual services to physical services. Physical services have more detailed information than virtual services. A service handle and related description are included for better execution. For each virtual service, workflow engine can select a suited physical service to do real computing and processing. At the same time, the middle results and data are transferred among these grid nodes where the physical services are located.

2.4. On-line Grid Programming

On-line Grid Programming provides a visual programming toolkit named Vimage for users. With this toolkit even users who are not familiar with programming can easily finish the fussy coding work. Further more, information about the application constructors are available for the real platform. Users only select needed application constructors, join them with connectors, and then make their own image grid programs. The structure of on-line Grid Programming is shown in Figure 6.

The Vimage system first checks the program to make sure that there are no lexical mistakes, and submits it. Users then are able to view their jobs and edit them.

Application Constructor List and Descriptions – Application constructor List shows a list of instantaneous application constructors and the services provided by ImageGrid. From the column of application constructor description, the specification and the parameters' type of application constructors can be obtained. This sub-module is based on the information service module, which makes the real-time display feasible.

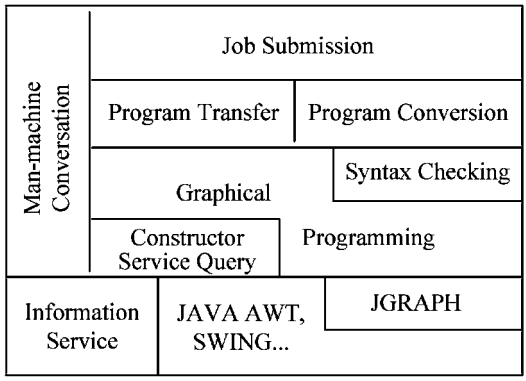


Figure 6. An overview Structure of On-line Grid Programming Graphical System

Graphical Programming – Graphical Programming sub-module presents an easy way to program grid applications. The coding work can be finished just like drawing. First, choose a service from the application constructor list; then, fill in the real parameters for the program sheet. When you finish, try another module that you need. Finally, connect these modules, save and submit them. After syntax checking by Vimage, the job will be processed to the Image Grid platform.

Job submitting – The sub-module of Job Submitting deals with the operations of job submitting. The dominating function of this part is the conversion from graphical program to the codes the platform can read and the transfer from clients to servers.

User Program Management – Users can view the status of their program submitted to the platform and delete the submitted program through User Program Management.

Statistics and Analysis – The module of statistics and analysis presents some statistics data from programs or the system itself.

3. Image Processing Applications

Here we introduce three typical image processing application in ImageGrid: digital virtual body reconstruction, remote-sensing image processing, medical image diagnoses.

3.1. Digital Virtual Body Reconstruction

3.1.1. Introduction and Requirements of Digital Virtual Body

VHP (Visible Human Project) was proposed by the National Library of Medicine of USA in 1993. In the following decade, many world-famous institutions carried out much research work based on VHP data and obtained tremendous achievement like image auto-segmentation, 3D reconstruction algorithm, image registration and so on. Using these new technologies, many advanced systems or models have been developed based on VHP dataset such as virtual endoscope, virtual patient model, etc.

To fit physical features of orient people, our country decides to develop Visible Chinese Human (VCH) system with our own copyright. Some institution published several representative VCH datasets successively from the beginning of 2003. In the field of obtaining dataset, our researchers have proposed some new advanced technologies and the-

ories and as the result, the quality of dataset is higher than that of other countries in some respects. On the other hand, however, in the field of dataset processing, surface reconstruction and result application, we still have great difference between these advanced countries.

The main process of realizing digital virtual body is getting the contour data of outer shape and inner organs through recognition and segmentation, reconstructing the triangle mesh of human body through Marching Cube [10] or other method. Commonly, because the obtained triangle mesh data is very huge and not good at visual effect, it should be optimized from smoothing and simplification operation. Connecting with some special background database and providing a friendly interface, the digital virtual body platform should be established.

Since the quantity of VCH dataset is very huge, e.g., the capacity of original dataset is over 20GB, the amount of calculation is numerous. When execute recognition and reconstruction on these data, the needed power of computing resources will be even higher. Fortunately, grid with robust computation and storage abilities can satisfy this requirement. To take full advantage of grid resources, obviously, all the algorithms must be improved in parallel mode and then, the digital virtual body system will be practical because user can complete various operations quickly.

3.1.2. Digital Virtual Body Reconstruction in ImageGrid

As a representative application in ImageGrid, the application of digital virtual body reconstruction provides some algorithmic services besides 3D reconstruction. In detail, the following algorithms will be available: mesh reconstruction, mesh smoothing and mesh simplification algorithms. Each service accepts the format-given data and required parameters via grid interfaces. If a mission is submitted, the system will search some available computing resources from ImageGrid automatically. After completing corresponding operations, the 3D results will be returned.

Remote visualization acts as a crucial part in 3D reconstruction. Usually it's used for generic users to view or browse long-range virtual scene or 3D model by remote high performance computers. Nowadays most of remote visualization system is implemented either by loading 3D file at client to render or else through the method raised by Bernardini. But both of them include such-and-such shortage, for examples, overlong download time, biggish network bandwidth and enormous computer powers, and so on.

In order to resolve these difficult problems, we introduce IBR (Image Based Rendering) technology into the tools of reconstruction and visualization in ImageGrid. As far as known, the conundrum of achieving full interactivity without losing visualization quality is thus far unanswered. But our method has given a chance to this problem. McMillan's 3D Warping algorithm is adopted and improved. The inputs to 3D warping are depth images, which are 2D color images containing depth information at each pixel. Each depth image also contains a viewing matrix with fame buffer and depth buffer. Comparing with the traditional 3D graphic pipeline, new 3D warping algorithm demands less computing powers.

C/S mode is used in remote visualization system. When the server receives interactive information periodically, it updates depth images based on the input parameters and transmits new depth images to the client. The communication between client and server is the compound of both synchronous and asynchronous visualization modes (Figure 7):

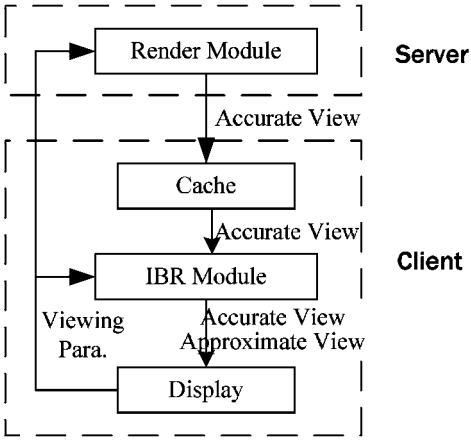


Figure 7. Compound Communication Mode of Synchronous and Asynchronous Mode

synchronous mode is for better visualization quality and asynchronous mode is for better remote visualization interactivity.

Firstly, user sends interactive information to server and client's IBR rendering module. The server renders 3D files to get depth images (called "accurate view") with synchronous mode. At the same time, IBR engine do an asynchronous with the corresponding work at the client to generate "approximate view". When user browses the 3D model or virtual scene rapidly he/she can watch the approximate view re-constructed by 3D warping method. But when the observer finds something attractive and wants to stop and examine the special view, the approximate view will be replaced by the accurate view built with real data from the server. With this method, the server also may take advantage of grid resources to reconstruct vivid 3D images, and the client can hide computing costs and network latency by performing 3D warping to update the display at interactive rates. Even when network is down or the server fails to update depth images, the client can still work in stand-alone mode. Meanwhile, users needn't tradeoff between the visualization quality and interactivity.

Figure 8 is the reconstruction of virtual body, liver and kidney based on Virtual Chinese Human II (VCH II) in ImageGrid.

3.2. Remote-sensing Image Processing

3.2.1. Introduction and Challenges of Remote-sensing Image Processing

Remote sensing is a kind of science and technology for measuring, analyzing and recognizing a target object's characteristics by means of some sensors or instruments without directly touching the object. Currently, the application of remote sensing technology is being developed from qualitatively to quantitatively, from statically to dynamically. Meanwhile, theories and technologies of remote sensing information processing closely related to remote sensing application have been substantially progressed. Remote-sensing image processing is performing digital processing with computer to enhance and obtain professional information from remote-sensing image.

With the rapid innovations of information and remote sensing technology, remote-sensing image processing is facing three big challenges: a great demand of huge storage

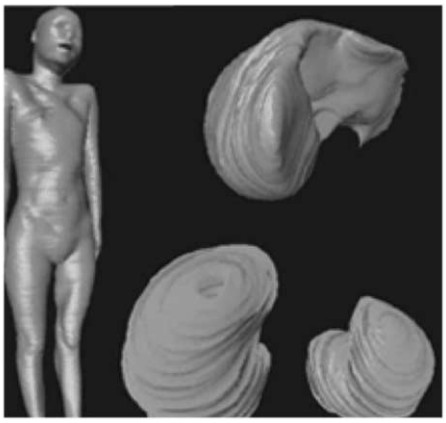


Figure 8. Reconstruction of virtual body, liver and kidney

resource, an increasing complexity of computation and an intensive requirement of fast even real-time processing. The size of the image, for example, is up to gigabytes; and only a process of geometric correction for one 10000*10000 image needs over ten billion float operations. Obviously, single processor system would not meet all these demands. Therefore, massively parallel processing (MPP) or high performance computing (HPC) becomes the best way to speed up the processing and analyzing of remote-sensing image. However, considering allowed cost of production, not all research organizations own costly supercomputers. On this background, how to utilize low cost computing resources and idle computing capacities in collaborative research environment receives much attention.

As a solution of cooperative computing, grid computing supports traditional parallel applications and aims to offer consistent and inexpensive access to resources irrespective of their physical locations or access points, which can settle these new challenges of remote-sensing image processing:

First, grid technologies can integrate rich computation and storage resources in computational grid. With these aggregated tremendous capacities, distributed super computing can be supported in larger scale.

Second, computational grid can support traditional parallel applications. For example, using MPI-G2, standard MPI programs can run transparently on various grid nodes such as clusters or heterogeneous computers located faraway.

Third, encapsulating remote-sensing image processing into grid service can facilitate more users, and the values of the applications will be fully presented. Additionally, grid can offer secure authentication mechanisms, provide dynamic and static information of resources, and construct virtual labs for users.

3.2.2. Parallel Remote-sensing Image Processing Service System in ImageGrid

Parallel Remote-sensing Image Processing Service System based in ImageGrid [11] (PRIPSS-G) puts PRIPS (Parallel Remote-sensing Image Processing System) software system into grid environment. Multifarious remote-sensing image processing parallel algorithms are provided as grid services for grid users, which cover the three phases of

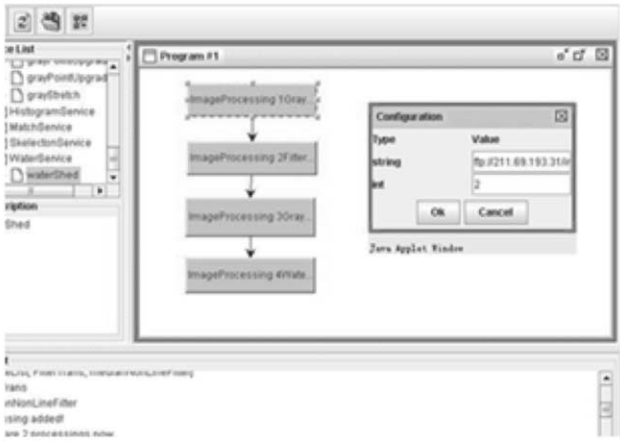


Figure 9. Grid application construction and submission

remote-sensing image processing [12]: preprocessing, basic processing and advanced processing, totalizing fourteen large sorts and thirty-five kinds.

Considering the primary efficiency and performance of remote-sensing image processing, all parallel algorithms are exploited with standard C language and MPI library is adopted for the parallelization. According to the standard of grid service, they are all encapsulated as services with Java programming. After deploying them into ImageGrid, users can obtain the capabilities of grid services and develop remote-sensing image processing applications to satisfy their own requirements.

ImageGrid utilizes the computational resources in grid and image processing solving environment to transmit remote-sensing image processing application to be workflow by the component and workflow techniques, which not only decreases the user's work hard, but also improves the efficiency and solves more matters by computational resources.

To illuminate remote-sensing image processing application in ImageGrid, we refer to take the landform recognition example of satellite remote-sensing images. It is composed of four operations: gray scale transformation of image enhancement, median filter of noise reduction, gradient transformation of edge enhancement and water shedding of image segmentation. The traditional method is to do a series of linear operation step by step, which makes heavy interactions and frequent communication / data transmit among grid nodes and managing center.

Users only construct the application and point out the logistic relationship and image data, the other things are processed automatically by the system: converting the application requirements to workflow description, picking out required service and integrating them into grid workflow for execution.

Figure 9 is a graphic interface for the application, through which application can be constructed via drag and drop. The real-time grid services and detailed descriptions are listed on the left. The right is user's main working panel, in which the relationships of service components are presented with service panes and arrows. The traces of user's actions are listed at the bottom. Grid service components are dragged from the service lists and dropped onto the right canvas.

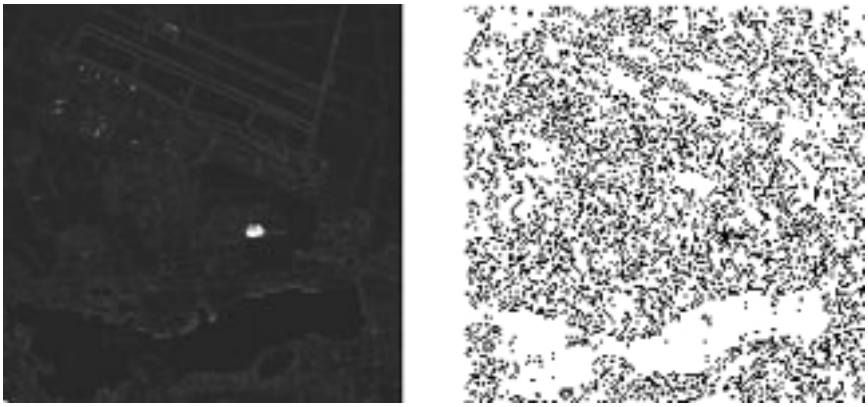


Figure 10. The result of the remote-sensing image processing example

The visual result of the example is shown in Figure 10. The source image is a picture of an airport with a lake aside, the target after four linear steps is on the right. Users can download the processed images on their own.

3.3. Medical Image Processing

3.3.1. Introduction and challenges of Medical Applications

With widely use of digitalized medical equipments such as CT (Computing Tomography), MRI (Magnetic Resonance Imaging), PET (Positron Emission Tomography) and DR (Digital Radiology) in clinical diagnosis, coupled with prevalence of computer technology in hospital, medical image data has been increased exponentially, and faces huge challenge to storage, computation and management.

The healthcare information systems such as HIS (Hospital Information System), PACS (Picture Archiving and Communication System) and CIS (Clinical Information System) are widely utilized for managing hospital data and basic communications in one hospital. But these medical systems are heterogeneous and incompatible, which adopt various computing platforms and data storage environments designed by different developers. This non-uniformity has given rise to the problem of how to integrate medical resources and data within or across hospitals.

The other challenge stands out as a result of exponential growth of medical image data and rapid development of medical imaging technology. Medical images with visual healthy or diseased features are very important to assist doctor's diagnosis. But the growth of medical data has increased the workload and enormous pressure to doctors' diagnosing. That means clinicians are drowning in data. Some rare or tiny details in images may be ignored regrettably, and only experienced clinicians can quickly find out abnormal images from mass image cases.

The combination of medical image processing and grid technology will arouse new approaches to solve these challenges. The aim of medical image grid is to make use the grid technology to integrate different medical systems among across-area hospitals, and turn into a clinical application environment to support information sharing, cooperative teamwork, and assistant diagnoses.

Figure 11. MedImGrid Information Integration

3.3.2. Medical Image Application in ImageGrid

Medical image application in ImageGrid (MedImGrid) [13] is apart from other computing-sensitive grid applications, which is a kind of data-sensitive application with the key problems of medical information integration, searches and medical image analysis and processing.

Medical Information Integration in MedImGrid is based on a distributed multi-agent architecture and uses HL7 [14] as accredited medical information exchanging standard [15,16]. A HL7 application middleware has been developed to access medical information systems through compatible HL7 or DICOM standards, which avoids the direct database operation. MedImGrid retrieves medical images and information from date agents on medical grid nodes through CGSP-DAI. With these data resources, medical integrating module implements seamless integration and uniform assess of medical information and images on different hospitals or platforms. Upper portal utilize user-context and metadata to discover grid services and retrieve information. Intelligent, humanized human-computer interaction interfaces are provided for various medical applications.

One medical information integration instance of decentralized data nodes is shown in Figure 11. It retrieves emphysema cased from different hospitals. Online image processing tool are also provides to open and process medical images with DICOM (Digital Image Archiving Communication System) format.

There are two kinds of medical image processing applications in MedImGrid. One is CSBIR (Content and Semantic Context based Image Retrieval) application, which can be used by clinics to find feature-like images to aid his diagnose according to image content and user context. Figure 12 is the instance of retrieve thorax CR images from distributed image libraries. The other is CT-based 3D reconstruction application. Figure 13 shows the reconstruction of 30 slices of head CT images based on Marching Cubes algorithms.

Within these two applications, all medical image processing algorithms are encapsulated as grid services and stored in grid algorithm barn. A metadata table is used to record the description of them, as the clues to find out the most suitable and optimal algorithm for various medical image processing applications. All chosen algorithms can be processed on available clusters or high-performance computers in MedImGrid.

Figure 12. Instance of medical image retrieval

Figure 13. 3D head reconstruction based on CT

4. Conclusion and Future Work

In this paper, some complicated problems in ImageGrid platform, middleware and tools are summarized, and image processing applications are tackled with the technologies of Grid. As a solution, complex grid applications are evolving into sets of cooperating components, which should be constructed as grid workflow by assembling these components in grid environment to get high performance in computing grid. The integrated grid architecture of image processing solving environment are discusses then. The significance and implementation of three typical image processing applications, digital virtual body reconstruction, remote-sensing image processing and medical image processing are introduced in the end.

Yet, no matter grid middleware or image processing applications in ImageGrid need further improving and expanding, and the researches are underway. There are many factors and aspects that should be thought carefully, such as the universality of remote-sensing image processing algorithms, the standardization and integration of medical information and data, the security about the transfers of medical data and the secrecy of patient's privacy. The support of more complex and more flexible processing of grid image processing and related grid modules also should be strengthened in the near future.

Acknowledgements

This work is sponsored by ChinaGrid project from China Ministry of Education under grant No.CG2003-GA001 and China Next Generation Internet (CNGI) project under grant No.CNGI-04-15-7A.

References

[1] I. Foster, C. Kesselman. *The Grid: Blueprint for a new computing infrastructure*. San Fransisco: Morgan Kaufmann Publisher, 2003.

[2] S. Hastings, T. Kurc, S. Langella, et al. Image Processing for the Grid: A Toolkit for Building Grid-enabled Image Processing Applications. In *Proceedings of 3rd International Symposium on Cluster Computing and the Grid*, 2003.

[3] R. Zheng, H. Jin, Q. Zhang, etl al. IPGE: Image Processing grid environment using components and workflow techniques. In *Proceedings of the 3rd International Conference on Grid and Cooperative Computing*, 2004.

[4] H. Jin, X.H. Shi, L. Qi. Use Case Study of Grid Computing with CGSP. In *Proceedings of International Conference on Human.Society@ Internet(HSI2005)*, 2005.

[5] H. Jin. ChinaGrid: Making grid computing a reality. In *Proceedings of International Conference of Asian Digital Libraries 2004*, 2004.

[6] I. Foster, C. Kesselman, J. Nick, et al. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. Open Grid Service Infrastructure WG, Global Grid Forum, 2004. Available as pdf file at http://www.globus.org/research/papers/ogsa.pdf.

[7] W. Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125-203, 2002.

[8] H. Jin, R. Zheng, Q. Zhang, et al. Components and Workflow Based Grid Programming Environment for Integrated Image Processing Applications.*Concurrency and Computation: Practice and Experience*, 18(14):1857-1869, 2006.

[9] K. Murphy. An Introduction to Graphical Models. Massachusetts: Intel Research, 7-11, 2001.

[10] M. Gregory. Nielson. On Marching Cubes. *IEEE Transactions on Visualization and computer Graphics*, 9(3): 283-297, 2003.

[11] R. Zheng, H. Jin, Q. Zhang, et al. Workflow-based Remote-Sensing Image Processing Application in ImageGrid. In *6th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2005.

[12] X.J. Yang, Z.M. Chang, H.F. Zhou, et al. Services for parallel remote-sensing image processing based on computational grid. In *Proceedings of the 3rd International Conference on Grid and Cooperative Computing*, 2004.

[13] H. Jin, A.B. Sun, Q. Zhang, et al. MIGP: Medical Image Grid Platform Based on HL7 Grid Middleware. In *Proceedings of Advances in Information Systems. 4th International Conference (ADVIS'06)*, 2006.

[14] P. Marcheschi, A. Mazzarisi, S. Dalmiani, et al. HL7 Clinical Document Architecture to Share Cardiological Images and Structured data in Next Generation Infrastructure. In *Proceedings of Computers in Cardiology*, 2004.

[15] D. Meo, D. Quarto. An HL7-aware Multi-agent System for Efficiently Handling Query Answering in an E-health Context. In *OTM Confederated International Conferences, CoopIS, DOA, CADA, and ODBASE 2006 Proceedings*, 2006.

[16] B. Orgun. HL7 Ontology and Mobile Agents for Interoperability in Heterogeneous Medical Information systems. *Computers in Biology and Medicine*, 36(7-8):817-836, 2006.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

The EGEE-II Project: Evolution Towards a Permanent European Grid Initiative[1]

Owen APPLETON[a], Bob JONES[a], Dieter KRANZLMÜLLER[b], and Erwin LAURE[a]
[a] *CERN, Geneva, Switzerland*
[b] *GUP,Joh. Kepler University, Linz, Austria*

Abstract. Enabling Grids for E-sciencE represents the worlds' largest multi-disciplinary grid infrastructure today. Co-funded by the European Commission, it brings together more than 90 partners in 32 countries to produce a reliable and scalable computing resource available to the European and global research community. At present, it consists of more than 200 sites in over 40 countries and makes more than 35,000 CPUs available to users 24 hours a day, 7 days a week. This article provides an overview of EGEE, its infrastructure, middleware, applications and support structures. From this experience, the current state of future plans will be explained, which is summarized under the term European Grid Initiative (EGI), and represents an emerging federated model for sustainable future grid infrastructures.

Keywords. Grid computing, e-Infrastructures, scientific computing, sustainability.

Introduction

Over the past decade, European states and the European Commission have invested heavily in grid technology. A report [1] by the GridCoord project in 2005 shows more than 100 million Euro per year of investment on a national basis without EU funding, in the years 2002 to 2006. The European Commission reported a spending of 275 million Euro over the same timeframe.

Initially, these efforts were driven by academic proof-of-concept and test-bed projects, such as the European Data Grid project [2], but have since developed into large-scale, production grid infrastructures supporting numerous scientific communities. Leading these efforts is a small number of large-scale flagship projects, mostly co-funded by the European Commission, which take the collected results of predecessor projects forward into new areas. Among these flagship projects, the EU Project EGEE (Enabling Grids for E-sciencE) is intended to unite thematic, national and regional grid initiatives in order to provide an infrastructure available to all scientific research in Europe and beyond in support of the European Research Area (ERA)[3].

[1] This work is co-funded by the European Commission through the EGEE-II project, contract number INFSO-RI-031688.

The project has met with considerable success within the first three years of its four-year programme, and is now exploring ways to develop sustainable infrastructures to ensure the long-term availability of its grid infrastructure to the larger research community.

This paper provides an overview of the EGEE project, currently in its second two-year phase under the name EGEE-II. Section 1 introduces EGEE-II and some of its characteristics in numbers. Section 2 defines the grid landscape in Europe and EGEE's role in incubating collaborating grid projects. This is followed by a description of EGEE's infrastructure in Section 3, an overview of EGEE applications in Section 4, and an outlook of the future sustainable grid infrastructure in Section 5. Concluding remarks end the paper in Section 6.

1. EGEE in a nutshell

The EGEE project has turned the vision of the seamless sharing of computing resources on an international scale into reality. Co-funded by the European Commission, it was launched on 1 April 2004 to establish a European grid infrastructure in support of the European Research Area.

The first two-year phase of EGEE ended successfully on 31 March 2006, exceeding almost all its goals. In its second phase, EGEE-II, which started on 1 April 2006, over 90 partners in 32 countries are organised in 13 federations to make up the project's consortium. On the global landscape, EGEE has expanded beyond Europe, establishing international relationships with groups in the US and Asia.

The primary goal of EGEE is to produce and maintain a production quality grid infrastructure. Beyond this, it aims to spread knowledge about the grid and its benefits to researchers and students in high energy physics, life and earth sciences, astrophysics, computational chemistry, fusion and other fields. In addition, it has generated interest from a wide spectrum of IT vendors and business applications.

As a main result, the project has constructed an infrastructure with more than 200 sites in over 40 countries, making it the world's largest multi-science grid infrastructure, offering a 24/7 service to its users. Currently, this infrastructure processes up to 100,000 jobs per day from eight scientific domains, ranging from biomedicine to fusion science. In total, over 200 Virtual Organizations (VOs) use the EGEE infrastructure.

To organize and integrate the disparate computational facilities belonging to a grid and to make their resources easily accessible to users, the project has assembled its own grid middleware, the gLite distribution [4]. The gLite middleware was re-engineered by the project from a range of best-of-breed middleware components to contain a full range of foundation services as well as support for field- and application-specific high level services.

Built on the pan-European network GÉANT, EGEE has significantly extended and consolidated the EGEE grid in its second phase, linking national, regional and thematic grid efforts, as well as interoperating with other grids around the globe. The resulting high capacity, world-wide infrastructure greatly surpasses the capabilities of local clusters and individual centres, providing a unique tool for collaborative compute-intensive science, so-called "e-Science".

2. The pan-European grid landscape

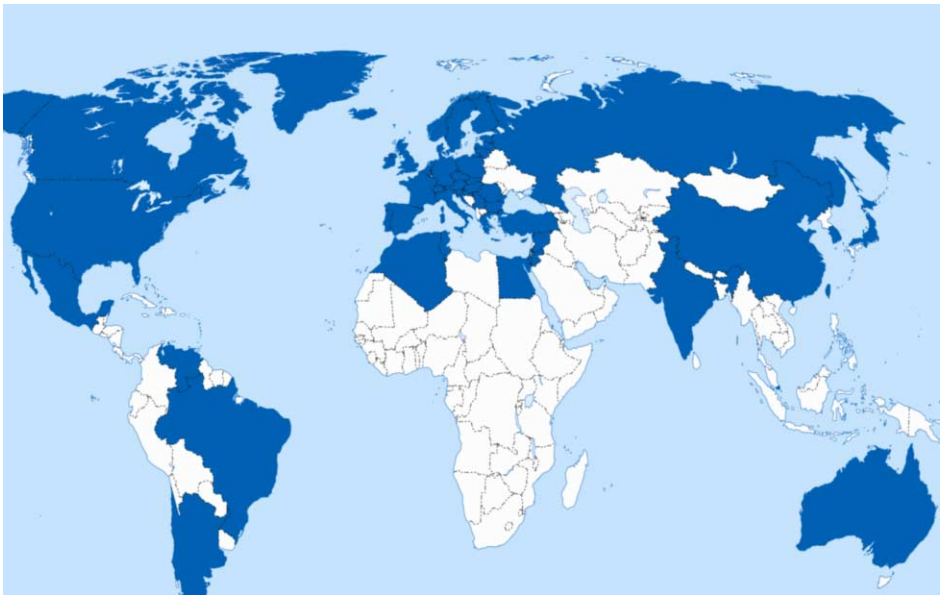


Figure 1. Countries within EGEE or connected to the EGEE Infrastructure via collaborating projects

The vision of grid computing that launched EGEE implies a sharing of resources across institutional, disciplinary and national boundaries, in contrast to 'enterprise grids' which often exist within individual companies. The broader vision pursued by EGEE and others requires members of individual grids to be aware of and cooperate with other related grid efforts to work toward interoperability at both an European and global level.

While EGEE primarily attracts computing centres with clusters of commodity PCs, the project is also collaborating with supercomputing grids via the DEISA (Distributed European Infrastructure for Supercomputing Applications) project, also funded by the European Commission.

Europe contains a number of projects, generally co-funded by the European Commission, which extend the reach of the EGEE infrastructure to new regions. These include European member states as well as other countries in areas such as the Baltic States, Latin America, China, India, the Mediterranean and South Eastern Europe (see Figure 1 and Table 1).

On the global stage, EGEE has close ties to Asia through partners in Taiwan and Korea (both of which operate sites on the EGEE infrastructure) and works with Japan's NAREGI project. In the US, EGEE again has project partners but also works with the Open Science Grid (OSG) and TeraGrid.

Both within Europe and world-wide, scientific research grid infrastructures face many of the same problems, and all are developing solutions for how to drive grid computing

PROJECT	COUNTRIES INVOLVED
BalticGrid	Estonia, Latvia, Lithuania, Poland, Sweden and Switzerland
EELA	Argentina, Brazil, Chile, Cuba, Italy, Mexico, Peru, Portugal, Spain, Switzerland, and Venezuela
EUChinaGrid	China, Greece, Italy, Poland, Switzerland and Taiwan.
EUIndiaGrid	India, Italy, United Kingdom
EUMedGrid	Algeria, Cyprus, Egypt, Greece, Jordan, Israel, Italy, Malta, Morocco, Palestine, Spain, Syria, Switzerland, Tunisia, Turkey and the UK
SEE-GRID	Albania, Bulgaria, Bosnia and Herzegovina, Croatia, Greece, Hungary, FYR of Macedonia, Moldova, Montenegro, Romania, Serbia, Switzerland, Turkey.

Table 1: Infrastructure projects connected to EGEE and deploying gLite

forward. One notable effort, organised through the Open Grid Forum (OGF) standards organisation, is Grid Interoperation Now (GIN)[5]. Through this framework, EGEE works with the other major infrastructures mentioned to make their systems interoperate with one another. Already this has led to seamless interoperation between OSG and EGEE, allowing jobs to freely migrate between the infrastructures as well as seamless data access using a single sign-on. Similar efforts are currently ongoing with NAREGI, DEISA, and NorduGrid. As part of the OGF GIN effort, a common service discovery index of nine major grid infrastructures world-wide has been created, allowing users to discover the different services available from a single portal [6].

3. The EGEE Infrastructure

The computing and storage resources that EGEE integrates are provided by a large and growing number of Resource Centres in all regions contributing to the infrastructure, coordinated by so-called Regional Operations Centres, or ROCs. The gLite middleware binds these resources into a single infrastructure to provide seamless access for the project's user communities. Users thus have access to resources that would not be available without the grid, permitting scientific investigations that would otherwise not be possible. Such a grid-empowered infrastructure facilitates collaborations of geographically dispersed communities who can also share data and resources.

In addition to the resources and services provided by EGEE, significant additional resources are made available to the research community through related infrastructure

projects and interoperation with grid infrastructures world-wide. A number of external groups from the research and business sectors also provide resources in order to gain experience as infrastructure providers. This makes up around 25% of the overall capacity of the EGEE grid infrastructure.

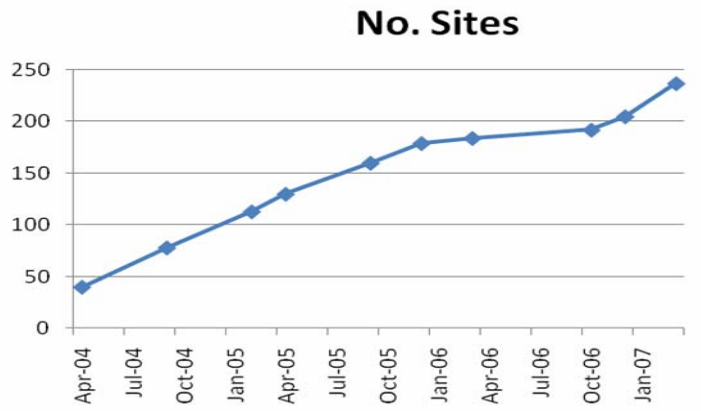
3.1. The EGEE Production Service.

The Production Service is a supported infrastructure capable of maintaining the high throughput of jobs needed to be of daily use to the scientific and research communities.

This production service is spread across more than 40 countries, managed by 10 Regional Operation Centres (ROCs), who share overall responsibility for the infrastructure on a rotating 8x5 basis. With the forthcoming addition of additional partners in the United States and Asia, this should move to a 24x7 worldwide coverage model in the future.

At present the production service has access to some 5 petabytes of storage and 35,000 CPUs, with numbers expected to rise dramatically over the next year as extra resources for the experiments of the forthcoming Large Hadron Collider (a next-generation particle accelerator currently under construction at CERN, Switzerland) and other new applications come online. Already these figures considerably exceed the goals planned for the end of the four-year EGEE programme, demonstrating the enthusiasm within the scientific community for EGEE and grid solutions. In the first year of EGEE-II, the production service processed 19.6 million jobs totalling 8400 CPU-years.

The distributed operation model used by the project is key to the success of EGEE—a centralized operational model would not be scalable to the large amount of sites federated in EGEE. It is worth noting that in 2006, EGEE saw a three-fold increase of the workload together with significant increases of the infrastructure both in terms of sites and CPUs without any impact on the overall operations of the infrastructure. Figure 2 shows the development of EGEE in terms of numbers of sites.



.

Figure 2. Development of the EGEE infrastructure

This service also interoperates with other major grids worldwide, in particular OSG in the United States. Work is also ongoing to interoperate with the DEISA project and the Advanced Resource Connector (ARC) provided by NorduGrid.

3.1.1. The Pre-Production service.

The EGEE Pre-Production Service runs across 27 sites, providing a limited environment for medium scale testing of new middleware releases and applications. Working in a truly distributed manner that as far as possible mimics the properties of the production service allows the project to test new software in the most rigorous way possible prior to deployment across the whole production service. This testing is vital for such a large grid, as any errors or faults in deployed software would not only expose many sites to security risks, but would also be time-consuming to fix due to the diverse properties and geographical spread of the grid sites connected to the EGEE infrastructure.

3.1.2. The GILDA testbed.

The project also features a specialized grid testbed dedicated to dissemination, training, porting and early stage testing. GILDA (the Grid INFN Laboratory for Dissemination Activities) was set up by EGEE partner INFN (Istituto Nazionale di Fisica Nucleare) in 2004 to aid in a wide range of activities. It allows members of the public or prospective new users of the EGEE infrastructure to try the grid in a simple but realistic way, including all the normal grid elements such as certification authorities, resource brokers, storage and computing elements.

Within the EGEE training activity, the GILDA testbed also functions as training infrastructure, which emulates the production e-Infrastructure but has additional properties that meet particular training and student needs – such as pre-production features, short response times, light-weight authorization and convenient portals. This allows trainees to learn about grid use in an environment as close as possible to the production service such that their transition to real grid usage is smooth. In addition to these training functions, GILDA is also used as a platform for accelerated application porting and testing, making use of the small testbed to "gridify" an application in a very short period of time. This service allows prospective or new users to gain understanding not only of the grid but of how their applications will be used on it.

3.2. The gLite Middleware

EGEE deploys the gLite middleware [4], a middleware distribution that combines components developed in various related projects, in particular Condor [7], the Globus Toolkit (GT) from Globus [8], LCG [9], and VDT [10], complemented by EGEE developed services. This middleware provides the user with high level services for scheduling and running computational jobs, accessing and moving data, and obtaining information on the grid infrastructure as well as grid applications, all embedded into a consistent security framework.

The gLite middleware is released with a business friendly open-source license (based on the Apache 2 license), allowing both academic and commercial users to download, install and even modify the code for their own purposes.

The gLite grid services follow a Service Oriented Architecture [11], which will facilitate interoperability among grid services and allows to easily complement the

services provided by gLite with application specific services, such as metadata catalogs and meta-schedulers, which is a common usage pattern on EGEE.

The focus of gLite is on so-called *foundation services*. These services provide the basic middleware infrastructure that defines the EGEE grid and encompass the security infrastructure, the information and accounting systems, as well as access to computing and storage resources, typically referred to as Compute Elements and Storage Elements. Examples of higher level services, like a meta-scheduler, replica catalog, or file transfer service are included in the gLite distribution as well. The gLite security infrastructure is based on X.509 certificates, involves sophisticated tools for VO management and local authorization and is currently being integrated with upcoming organization membership services based on Shibboleth [12]. The information system is based on hierarchical 'ldap' servers and the accounting system uses the OGF defined usage records facilitating interoperability with other infrastructures. Access to compute resources is provided via Condor and GT2 GRAM (which is currently being updated to GT4 GRAM), work on a HPC-Profile/BES [13] compliant interface is ongoing [14] and EGEE has standardized on the SRM [15] interface to storage resources, for which a number of production-level implementations are available. GridFTP is used for data transfer.

The gLite distribution is available from the gLite webpage: http://www.glite.org. The current version 3.0 was released in May 2006 and will be upgraded to 3.1 during summer 2007. More details on gLite are available in [4] and via the gLite documentation available from the gLite webpage.

EGEE has taken a conservative approach in defining the gLite composition, avoiding frequently changing cutting-edge software while tracking emerging standards. For a production infrastructure, reliability and scalability are of higher value than the exploitation of the very latest advances that still need time to mature. EGEE is moving towards web services adhering to WS-Interoperability recommendations wherever possible.

In order to achieve the required reliability and scalability to support the size and workloads of EGEE a strict software process, involving several stages of testing has been applied. This process, which has been modelled after industry strength processes, includes software configuration management, version control, defect tracking, and an automatic build system. The experiences gained with the gLite developments are made available to other software projects via the ETICS project [16] that offers the software development tools to a wider community.

4. EGEE Applications

EGEE actively engages with application communities, beginning with High Energy Physics and Biomedicine at the start of EGEE but expanding to support many other domains. These include Astronomy, Computational Chemistry, Earth Sciences, Financial simulations, Fusion science and Geophysics, with many other applications currently being evaluated by the project.

The applications running on the EGEE Grid are rapidly moving from testing to routine usage, with some communities already running large numbers of jobs on a daily basis. These communities are organized in Virtual Organizations, groups of people working in the same domain or specific research area. Virtual Organizations allow users to access common data sets and tools, and enable remote collaborations in

research. Already, EGEE supports more than 200 such Virtual Organizations. These include both groups actively supported by EGEE, with help being provided for application porting and the inclusion of special features into the gLite middleware, and groups who simply make use of the general services EGEE provides. The following sections highlight two of the application areas supported by EGEE.

4.1. Large Hadron Collider

Support for processing data coming from the forthcoming Large Hadron Collider (LHC), a next-generation particle accelerator under construction at CERN, Switzerland, was a major driver in the development of the EGEE programme. The LHC experiments are predicted to produce on the order of 15 petabytes of data per year, implying a scale of data-processing that made grid technology a natural choice. EGEE works closely with the Worldwide LHC Computing Grid (WLCG) collaboration set up to distribute and process this data. Through the EGEE infrastructure, members of the global physics community will be able to collaborate on the analysis of this data, which it is hoped will find the Higgs boson, an important step in the confirmation of the so-called 'standard model' in particle physics.

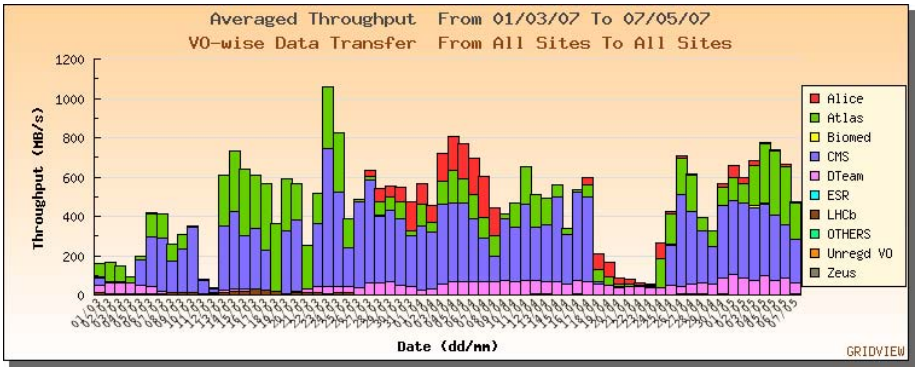


Figure 3. Average throughput for High Energy Physics experiments

These LHC experiments also help to stress-test the EGEE infrastructure due to their large-scale requirements. An example of this is the network bandwidth needed for the distribution of the LHC data, where the experiments have each demonstrated capacity of one petabyte per month transfer, and aggregate speeds for the LHC experiments have reached one gigabyte per second with real workloads (see Figure 3).

4.2. The WISDOM initiative

WISDOM, Wide In Silico Docking On Malaria, was launched in 2005 to use emerging information technologies to search for drugs for Malaria and other so-called 'neglected' diseases. WISDOM works closely with EGEE, and has made use of the EGEE infrastructure to run a number of large scale 'data challenges'. These are tests that screen large scale databases of molecules for potential treatments for disease. The first

of these was carried out in summer 2005, when 42 million compounds were screened for efficacy against a malarial protein in just 6 weeks. This was followed by a data challenge looking for potential treatments for the H5N1 Avian influenza, and a second challenge on malaria in winter 2006/2007. This challenge processed up to 80,000 compounds per hour, totalling 140 million compounds tested in the 4 months of the challenge. Furthermore, the grid enabled testing is much more cost effective than testing in research laboratories. WISDOM estimate the total cost of computing resources, power and time of running such a study on the Grid as €3.6 million[2] on the grid rather than some €350 million in a lab.

5. Towards a European Grid Initiative

While EGEE's success has been clearly demonstrated above, its status as a series of short term projects raises the question of the future of the infrastructure it has built. Already today, many scientific applications depend on the EGEE production grid infrastructure, new scientific collaborations have been formed thanks to the advancements of grids, and business and industry are getting very interested. However, a long term plan for the availability of the grid service model is needed to protect the investments of user communities.

This is of particular relevance as the series of EGEE projects are based on typical EU funding cycles, and the current phase of EGEE will last only until April 2008. This contradicts many of the needs of new users joining the grid community.

New users require some amount of investment, both monetary and in time, in order to enable their particular application for usage on the grid. This means that if a new user wants to use the grid, there is a certain barrier to be overcome before the grid can be exploited, and it is obviously necessary that these investments are protected. In the case of the grid, this means that the effort spent on getting the application grid-enabled must be exploited as much as possible. If, however, a grid infrastructure can only be guaranteed to be operational for a limited amount of time, then the user community needs to decide whether the effort required to get an application started on the grid is justified. This is a clear limitation of the cyclic project-funding used by EGEE and other similar efforts, where a subsequent project phase is always determined by a corresponding proposal application process. A solution to this situation is urgently needed for Europe to maintain its leading position in global science grids, and to ensure a reliable and adaptive support for all sciences with the ERA.

As a consequence, EGEE has performed a series of activities together with national European grid efforts to establish the production grid infrastructure as a sustained, permanent research tool for science and engineering.

The initial outcome of these discussions is an operational model based on National Grid Infrastructures (NGIs) coordinated by a body tentatively named the European Grid Initiative (EGI). Driven by the needs and requirements of the research community, it is expected that this setup will enable the next leap in research infrastructures.

[2] EGEE does not pay for this work, those costs include the equipment the tests are run on (which are donated to the project by the partner institutes), and the time of members of EGEE and WISDOM that is funded by the EC or donated by the WISDOM members.

Based on the experiences and building on the results achieved in EGEE described above as well as related national and European efforts, the objectives of the future NGI/EGI organization should be to [17]:

- Ensure the long-term sustainability of the European e-Infrastructure;
- Coordinate the integration and interaction between National Grid Infrastructures;
- Operate the European level of the production grid infrastructure for a wide range of scientific disciplines to link National Grid Infrastructures;
- Provide global services and support that complement and/or coordinate national services (Authentication, VO-support, security, etc.);
- Advise National and European Funding Agencies in establishing their programmes for future software developments based on agreed user needs and development standards;
- Coordinate middleware development and standardization to enhance the infrastructure by soliciting targeted developments from leading EU and national grid middleware development projects;
- Integrate, test, validate and package software from leading grid middleware development projects and make it widely available;
- Provide documentation and training material for the middleware and operations. (National Grid Infrastructures may wish to make the material available in turn in their local language);
- Take into account developments made by national e-Science projects which were aimed at supporting diverse communities;
- Link the European infrastructure with similar infrastructures elsewhere;
- Promote grid interface standards based on practical experience gained from grid operations and middleware integration activities, in consultation with relevant standards organizations;
- Collaborate closely with industry as technology and service providers, as well as grid users, to promote the rapid and successful uptake of grid technology by European industry.

A major part of the work envisaged here will be carried out by National Grid Infrastructures. EGEE plays a major role in establishing these NGIs by providing the nucleus for the tasks mentioned above, for instance through its distributed operational infrastructure, and transferring the knowledge gained by operating a large scale production infrastructure. However, from an organisational point of view NGIs are not yet fully operational today. In fact, while most countries have established NGIs to enable the construction and operation of a national grid, the diversity between these countries is very high. They range from single individuals who act as a point of contact for the national grid community to operational national grid infrastructures; a wide variety of levels of maturity can be observed.

It is therefore essential that EGI supports the development and progress of NGIs as much as possible, that EGI defines the workload distribution between the individual NGIs and the EGI together with the appointed NGI representatives, and that the NGI representatives drive the establishment of the EGI organisation.

Another contributor to the EGI is the e-Infrastructure Reflection Group (eIRG), which has provided a series of important actions as ground work for the development of a sustainable grid infrastructure with a number of important steps.

- During the UK Presidency of the European Union in 2005, a European Vision for a universal e-Infrastructure has been formulated as follows [18]: "An environment where research resources (H/W, S/W & content) can be readily shared and accessed wherever this is necessary to promote better and more effective research."
- During the Austrian Presidency of the European Union in 2006, this statement has been further strengthened with the setup of the e-IRG Task Force on Sustainable e-Infrastructures (SeI), which formulated 5 recommendations in a corresponding report [19]. The report has been endorsed by the e-IRG Delegates and distributed to the European Commission and the European governments and funding agencies.

As a direct result of these efforts, a proposal for an EGI Design Study (EGI_DS) has been put together and submitted to the European Commission in May 2007. This will evaluate use cases for the applicability of a coordinated effort, identify processes and mechanisms for establishing an EGI, define the structure of a corresponding body, and ultimately initiate the construction of EGI.

This EGI_DS proposal has the unanimous support of the NGI representatives [20] and is currently under evaluation, while the EGI itself is continuing to seek support from the NGIs. At present, EGI is supported by all EU27 countries and 8 additional European countries, as well as partners from Asia, Latin America, and the US.

One of the major technical challenges will be the provision of a continuous Grid infrastructure during the transition from operations funded by projects like EGEE and others to the NGI/EGI structure. EGEE is preparing for that through its decentralized model which will be further refined in the coming two years and we expect a significant overlap time of both structures to ensure a smooth transition.

The above clearly indicates that Europe is on the verge of a new organizational form for coordinating and operating a sustainable grid infrastructure.

6. Conclusions

Since the launch of the EGEE programme in 2004, the project has made strides forward in both delivering a world-class production grid infrastructure and strengthening the grid field in general. Through provision of a reliable, scalable production infrastructure, high quality middleware and well developed support services, EGEE has attracted a wide range of users from the research and business communities.

These users began by experimenting with grid technology via EGEE, but are now shifting to daily use of the EGEE grid. This is particularly notable with the two original applications groups associated with EGEE, High Energy Physics and the Biomedical field, but these groups are rapidly being joined by groups such as Earth Sciences as major grid users.

While the user base has grown rapidly, with more than 200 Virtual Organizations now making use of the EGEE grid, the short project lifespan of EGEE and EGEE-II is becoming a limiting factor in EGEE's ability to attract users. To address this issue,

EGEE has worked alongside the European Commission and other members of the grid community to work on a long-term solution, namely the European Grid Initiative. This body would coordinate European grid computing in collaboration with the many emerging National Grid Initiatives to provide for long term support for grid users in Europe. Such a move will ensure that Europe keeps its place as the world leader in grid computing.

References

[1] GridCoord, "Deliverable 3.1.1 – Survey of Funding Bodies and Industry", EU Project Number IST-2003-511618, http://www.gridcoord.org/grid/portal/information/public/D.3.1.1_V.2.2_221105.doc (March 2005)

[2] F. Gagliardi, B. Jones, and E. Laure. "The EU DataGrid Project: Building and Operating a large scale Grid Infrastructure". In B. Di Martino, J. Dongarra, A. Hoisie *et al*. editors, *Engineering the Grid: Status and Perspective*. American Scientific Publishers, January 2006.

[3] For details see http://ec.europa.eu/research/era/index_en.html.

[4] E. Laure, S. Fisher, A. Frohner, *et al.*. "Programming the Grid with gLite" in, *Computational Methods in Science and Technology*, 12(1):33-45, 2006.

[5] For details see https://forge.gridforum.org/sf/projects/gin

[6] For details see http://forge.gridforum.org/sf/wiki/do/viewPage/projects.gin/wiki/GINInfoWiki

[7] D. Thain, T.Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience" *Concurrency and Computation: Practice and Experience*, *Vol. 17, No. 2-4, pages 323-356, February-April, 2005*

[8] I. Foster. "Globus Toolkit Version 4: Software for Service-Oriented Systems". *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, *pp 2-13, 2005*.

[9] For details see http://lcg.web.cern.ch/LCG/

[10] For details see http://vdt.cs.wisc.edu/

[11] D. Sprott and L. Wilkes. Understanding Service-Oriented Architecture. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmaj/html/aj1soa.asp.

[12] For details see http://shibboleth.internet2.edu/

[13] For details see https://forge.gridforum.org/sf/projects/ogsa-bes-wg

[14] For details see http://grid.pd.infn.it/cream/

[15] A. Shoshani, A. Sim and J. Gu. "Storage Resource Managers - Essential Components for the Grid". *In J. Nabrzyski, J. Schopf, J. Weglarz, editors, Grid Resource Management State of the Art and Future Trends Kluwer, 2003.*

[16] M. Begin, G. Sancho, A. Di Meglio, et al., "Build, Configuration, Integration and Testing Tools for Large Software Projects: ETICS", In *Proc. RISE 2006, International Workshop on Rapid Integration of Software Engineering techniques*, *13-15 September, 2006, University of Geneva, Switzerland, In Springer Verlag Lecture Notes in Computer Science (LNCS) Series, LNCS 4401, 2007*

[17] EGI Prep Team, "The Future European Grid Infrastructure – Towards a Common Sustainable e-Infrastructure", Vision Paper prepared for the EGI Workshop, Munich, Germany, http://www.eu-egi.org/public/EGI Vision (v1.1).pdf (February 2007).

[18] M. Read, "A European Vision for a Universal e-Infrastructure for Research", e-IRG (e-Infrastructure Reflection Group, http://www.e-irg.org/meetings/2005-UK/A_European_vision_for_a_Universal_e-Infrastructure_for_Research.pdf, London, UK (December 2005).

[19] e-IRG "Report of the e-Infrastructure Reflection Group Task Force on Sustainable e-Infrastructures", http://www.e-irg.org/publ/2006-Report_e-IRG_TF-SEI.pdf, Vienna, Austria (June 2006).

[20] EGI Prep Team, "EGI Letters of Support", http://www.eu-egi.org/los.php (May 2007).

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Innovative Grid Technologies Applied to Bioinformatics and Hurricane Mitigation

Rosa BADIA [a] Gargi DASGUPTA [b] Onyeka EZENWOYE [c] Liana FONG [d]
Howard HO [e] Sawsan KHURI [f] Yanbin LIU [d] Steve LUIS [c] Anthony PRAINO [d]
Jean-Pierre PROST [g] Ahmed RADWAN [f] Seyed Masoud SADJADI [c]
Shivkumar SHIVAJI [e] Balaji VISWANATHAN [b]
Patrick WELSH [h] Akmal YOUNIS [f]

[a] *Barcelona Supercomputing Center, Barcelona, Spain*
[b] *IBM India Research Laboratory, NewDelhi, India*
[c] *Florida International University, Miami, FL*
[d] *IBM T.J. Watson Research Center, Yorktown Heights, NY*
[e] *IBM Almaden Research Center, San Jose, CA*
[f] *University of Miami, Miami, FL*
[g] *IBM Products and Solutions Support Center, Montpellier, France*
[h] *University of North Florida, Jacksonville, FL*

Abstract. The Latin American Grid (LA Grid) joint research program fosters collaborative research across eleven universities and IBM Research with the objective of developing innovative grid technologies and applying them to solve challenging problems in the application areas of bioinformatics and hurricane mitigation. This paper describes some of these innovative technologies, such as the support for transparent to the application expert grid enablement, meta-scheduling, job flows, data integration, and custom visualization, and shows how these technologies will be leveraged in the LA Grid infrastructure to provide solutions to pharmagenomics problems and hurricane prediction ensemble simulations.

Keywords. Meta-scheduling, job flows, data integration, transparent grid enablement, custom visualization, bioinformatics, hurricane mitigation.

Introduction

Since December 2005, IBM has been engaged with academic partners in Florida, Puerto Rico, Mexico, Argentina, and Spain in the Latin American (LA) Grid initiative to dramatically increase the quantity and quality of Hispanic Technical Professionals entering the Information Technology fields.

At the core of this initiative is the development of a computer grid across multiple universities and businesses that serves as the platform for education and collaborative research in the critical and emerging fields of grid computing, distributed systems and supercomputing. The LA Grid constitutes a living laboratory for advanced research by the universities and IBM Research in application areas such as bioinformatics, hurricane mitigation, and healthcare.

In our ongoing research efforts, we aim to address application area problems with current state-of-the-art grid solutions by employing a top-down approach that provides the right level of abstraction for the domain experts while factoring out the common services that can be reused across domains. Among these common services, our focus has been on providing support for transparent grid enablement, meta-scheduling, job flows, data integration, and custom visualization. In this paper, we first describe the innovative technologies we developed in order to provide such support, and then we illustrate how these technologies can be leveraged towards the resolution of challenging problems in the area of bioinformatics and hurricane mitigation.

The paper is structured as follows. In Section one, we introduce the concepts of grid superscaling and transparent adaptation and show how these concepts can be combined to provide transparent grid enablement. In Section two, we detail our design for meta-scheduling and our on-going prototyping activities in that space. In Section three, we outline our approach to job flows, leveraging WS-BPEL and providing support for fault-tolerant job flows through a wrapping layer. In Section four, we describe the set of services we have been developing and the architecture we designed to provide data integration capabilities in grid environments. In Sections five and six, we show how our innovative technologies are being applied to address challenging scenarios in bioinformatics and hurricane mitigation. Section seven concludes this paper with our future plans towards the creation of a transparent grid environment, which will allow domain experts to express application logic using an appropriate visual interface while making transparent, to the greatest extent possible, the details of the grid hardware and middleware stack.

1. Transparent Grid Enablement

The advent of cluster and grid computing has created a remarkable interest in high performance computing (HPC) both in academia and industry, especially as a solution to complex scientific problems (e.g. bioinformatics and hurricane mitigation applications). To efficiently utilize the underlying HPC facilities using the current programming models and tools, however, scientists are expected to develop complex parallel programs; a skill that they might not necessarily have and is better done by HPC experts.

Current standards for cluster and grid programming such as MPI [1], OGSA [2], and WSRF [3] (and their implementations in offerings like MPICH2 [4], the Globus Toolkit [5], Unicore [6], and Condor [7]; to name just a few) have provided scientists with higher levels of abstraction. Noteworthy, these approaches have been successful in hiding the heterogeneity of the underlying hardware devices, networking protocols, and middleware layers from the scientist developer. However, the scientist is still expected to develop complex parallel algorithms and programs. Moreover, as the code for parallel algorithms typically crosscuts the code for business logic of the application, the resulting code is entangled and is difficult to maintain and evolve.

In this part of our research, we address these problems by enabling a separation of concerns in the development and maintenance of the non-functional aspects (e.g. the performance optimization) and the functional aspects (i.e. the business logic) of scientific applications. We achieve this goal by integrating two existing programming tools, namely, a Grid framework, called *GRID superscalar* [8], and an adaptation-enabling tool, called *TRAP/J* [9]. On one hand, GRID superscalar enables the

development of applications for a computational grid by hiding details of job deployment, scheduling, and dependencies and enables the exploitation of the concurrency of these applications at runtime. On the other hand, TRAP/J supports automatic weaving of alternative parallel code (including the corresponding calls to GRID superscalar runtime) into the sequential code developed by the scientist to support static and dynamic adaptation to heterogeneous grid computing environments.

1.1. Overview

Inspired by the *superscalar* processors, GRID superscalar provides an easy programming paradigm for developing parallel programs [8]. Similar to superscalar processors that provide out-of-order and parallel execution of machine instructions by bookkeeping their dependencies, GRID superscalar provides parallelism to the functions of a program written in a high-level programming language such as Java. Using GRID superscalar, a sequential scientific application developed by a scientist is dynamically parallelized in a computational Grid. GRID superscalar hides the details such as resource mapping, staging input data files, cleaning temporary data files, deploying and scheduling tasks, exploiting instruction-level parallelism, and exploiting data locality. We note that for many of its responsibilities, GRID superscalar depends on other grid computing toolkits such as GT4, Condor, and others.

TRAP/J is a tool that enables static and dynamic adaptation in Java programs at startup and runtime, respectively [9]. It consists of two GUI-based interactive tools as follows: (1) the *Generator*, which generates an adapt-ready version of an existing application by inserting generic hooks into a pre-selected subset of classes in the application, called *adaptable* classes; and (2) the *Composer,* which allows insertion of new code at the generic hooks both at startup or runtime. Adaptable behavior is provided through alternative implementations of adaptable classes. To replace alternative parallel algorithms developed using the GRID superscalar codes, we use the Generator to make the classes with sequential code adaptable, and then we use the Composer to weave in the parallel code.

Each tool provides us with the necessary features for transparent software adaptation from a sequential code to a grid-enabled one. Figure 1 illustrates the operation of our transparent grid enablement framework in the context of a simple case study, during which a sequential matrix multiplication program (developed in Java) is transparently adapted to run in a grid computing environment. First, we use GRID superscalar to develop alternative hyper-matrix multiplication algorithms by splitting the original matrices into a number of sub-matrices or blocks (Figure 1 (a), development time). Therefore, instead of just one task as in the original approach, using hyper-matrix multiplication and GRID superscalar, up to 4 tasks can be active at the same time. Of course, if we split the matrix into 9 blocks, then up to 9 tasks can be executed at the same time and so on and so forth. Next, using TRAP/J and GRID superscalar code generators, an adapt-ready version of the application is generated (Figure 1 (a), compile time). Next, a system administrator (or an intelligent software agent) configures the application to use the appropriate parallel algorithm based on the availability of resources, for example, the number of available nodes (Figure 1 (b), startup time). Finally, the GRID superscalar code—woven into the application using TRAP/J—will exploit the task-level parallelism by resolving the dependencies of the tasks, each performing multiplication of sub-matrices accumulatively (Figure 1 (b), runtime).

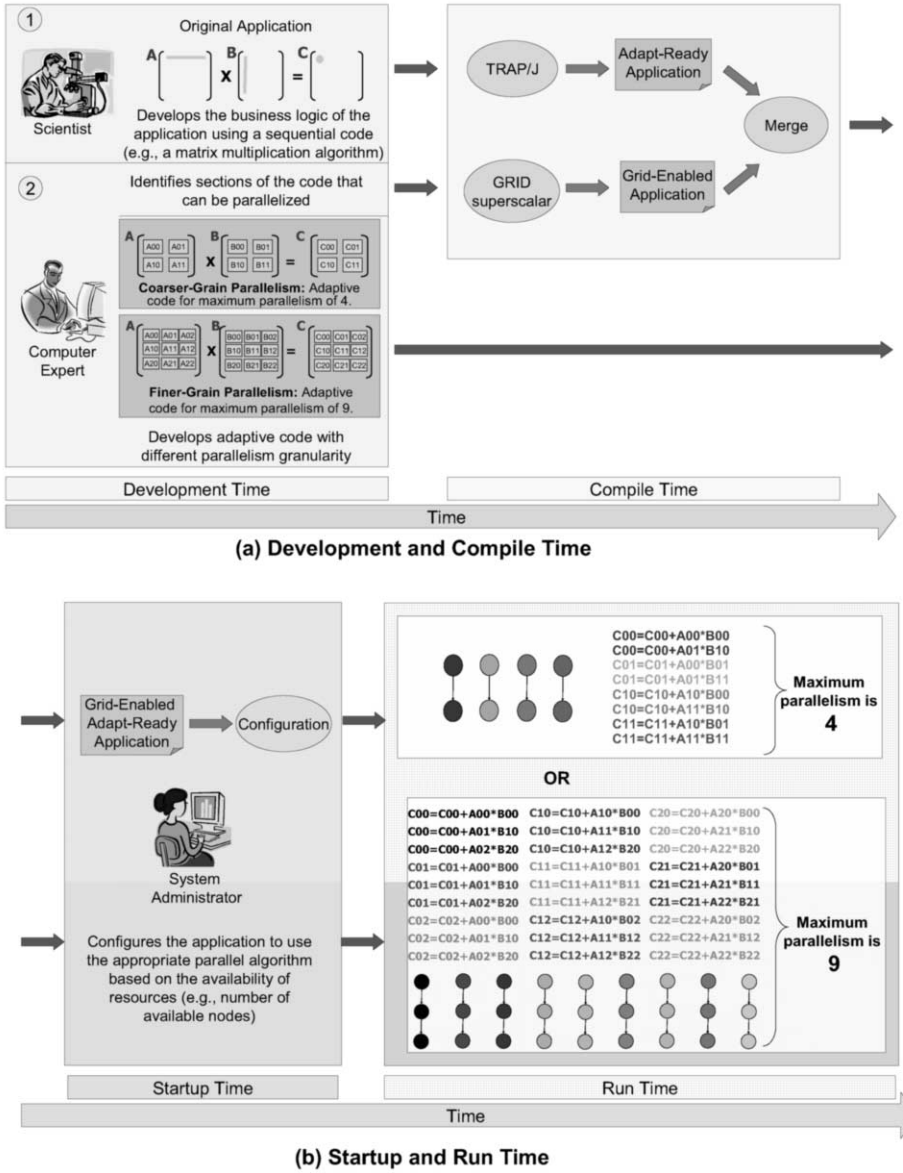
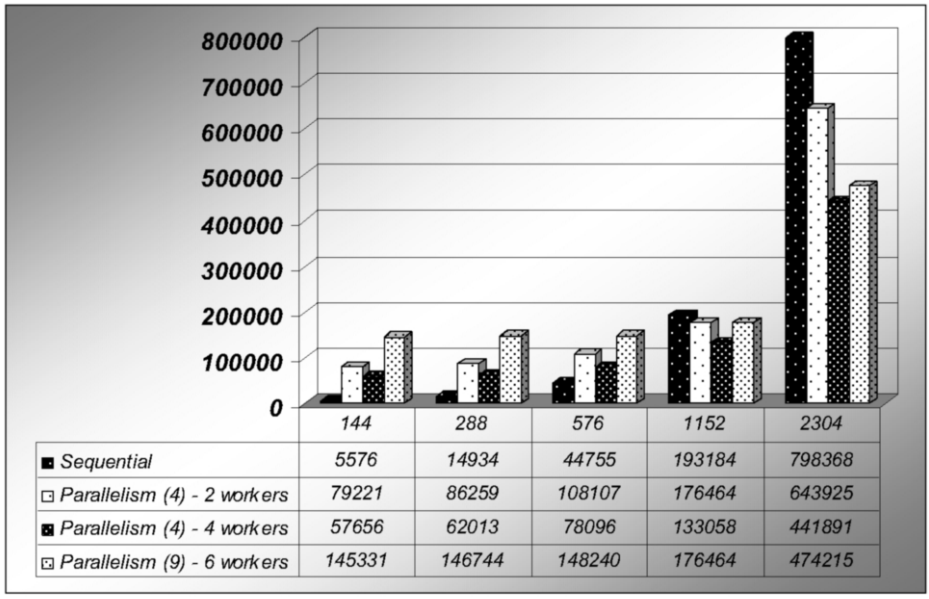


Figure 1. Grid enablement approach for the matrix multiplication case study

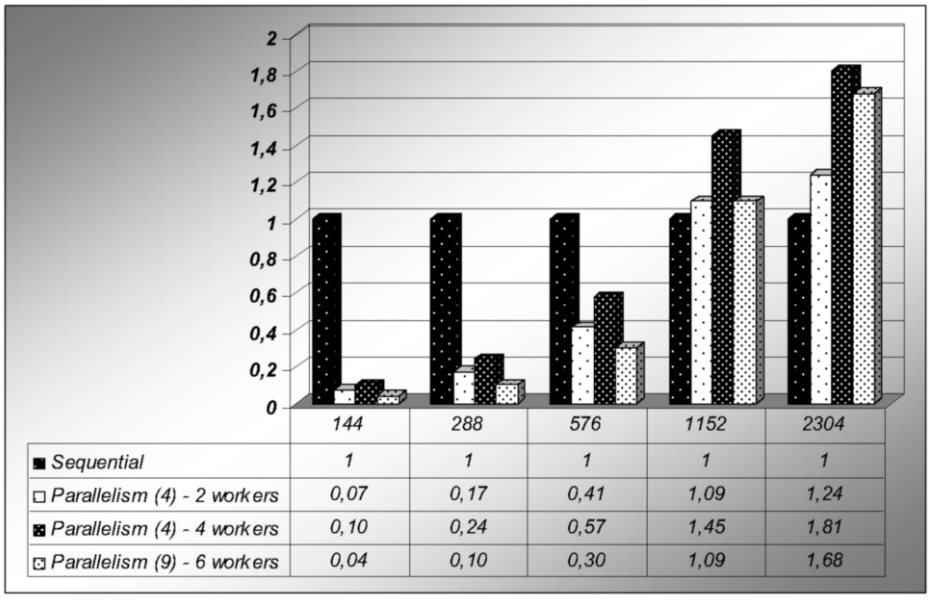
1.2. Experimental Results

To show the validity of our approach, we conducted a set of experiments that measure the speedup gained as a result of the grid enablement of the matrix multiplication application. The results are illustrated in Figure 2.



	144	288	576	1152	2304
Sequential	5576	14934	44755	193184	798368
Parallelism (4) - 2 workers	79221	86259	108107	176464	643925
Parallelism (4) - 4 workers	57656	62013	78096	133058	441891
Parallelism (9) - 6 workers	145331	146744	148240	176464	474215

(a) Execution time



	144	288	576	1152	2304
Sequential	1	1	1	1	1
Parallelism (4) - 2 workers	0,07	0,17	0,41	1,09	1,24
Parallelism (4) - 4 workers	0,10	0,24	0,57	1,45	1,81
Parallelism (9) - 6 workers	0,04	0,10	0,30	1,09	1,68

(b) Speedup

Figure 2. Execution time and speedup of grid-enabled versions of the matrix multiplication application

In Figure 2, we see that as the matrix size increases, the speedup improves and when the size of the matrix reaches 1152 (number of rows = number of columns = 1152), all the algorithms for the grid-enabled application perform better than the original sequential application. We notice that the algorithm, which uses 4 blocks on 4 worker nodes, exhibits the best performance. For a matrix of size 2304, it performs

almost twice as fast as the sequential application. This is because of the even distribution of the load and a one-to-one mapping of the blocks onto the worker nodes, which result in more parallelism and less communication overhead related to the file transfer between the nodes required both at the initialization and finalization stages.

We emphasize that these experiments are part of our ongoing research activities and they are not meant to be representative nor conclusive with respect to providing a quantitative metric for speedup gained because of the grid enablement. The main purpose of these experiments is to show that we were able to use our current transparent grid enablement prototype to transparently adapt an application to run in a grid computing environment.

1.3. Related Work

Satin [10] is a Java based programming model for the Grid which allows for the explicit expression of divide-and-conquer parallelism. HOCS [11] is a component oriented approach based on a master-worker schema. ASSIST [12] is a programming environment that aims at providing parallel programmers with user-friendly, efficient, portable, fast ways of implementing parallel applications. ProActive [13] is a Java grid middleware library for parallel, distributed and multi-threaded computing. Unlike our transparent grid enablement framework, none of the above mentioned approaches provide an explicit separation of concerns identifying separate tasks for scientist developers and HPC expert developers. Our framework can be extended to use these methods instead of or in complement to GRID superscalar and could be used as an enabler for supporting interoperation among the above mentioned approaches.

1.4. Future Work

As we mentioned before, we have been able to achieve *static* adaptation. Our next task will be to extend our framework in support of more autonomic behavior and include adaptation at runtime (dynamic) in response to high level system policies such as the addition of more nodes to the grid, a change in process scheduling or application level policies such as different blocking algorithms or faster algorithms. At present, dynamic adaptation of Java programs with TRAP/J is under progress and is being tested. Furthermore, moving towards building a more autonomic self adapting and self configuring system, we can expand our framework to provide context-aware adaptation, by keeping track of the state of the runtime environment and retrieve information about resource allocation, scheduling, etc.

2. Meta-scheduling

Over the past two decades, computing power has become increasingly available. However, the demand for computing power, driven by many new applications in bioinformatics, healthcare, physical science simulation, supply chain modeling, and business intelligent decision, still surpasses the supply. Grid computing allows harnessing of available computing resources from cooperating organizations or institutes, in the form of virtual organizations (VOs), in order to satisfy user demands and share the cost of ownership of the resources.

The first generation of grid technologies and infrastructures focused on harnessing the computational power of machines. With the evolution and availability of grid infrastructures, today's grid technologies further enhance the collaboration of users by providing easy access to greater varieties of resources, such as data and software services. For examples, the collection of astronomy data at one observatory can be easily made available to scientists around the world, the BioMOBY web services in Taverna [14] are used to support publishing and extracting of biological data.

At the core of grid technology is a resource brokering component, commonly known as meta-scheduler or meta-broker. The meta-scheduler matches user work requests to appropriate resources in the VO to execute the requests. In addition to the challenges of managing VO resources that have dynamic availability attributes, meta-schedulers need to take into account the resource usage policies and security restrictions enforced by the local schedulers, which they interact with.

Currently, there are many studies and systems related to meta-scheduling in the grid community. As discussed by Subramani et al. [15], most meta-schedulers can be classified into three different models:

- Centralized model: one meta-scheduling component has direct information of all resources available at the various institutes of the virtual organization and is responsible for scheduling job execution on all resources; local schedulers at individual institutes will act as job dispatchers. An exemplary system of this model is eNANOS [16].
- Hierarchical model: one meta-scheduling component has no direct access to resources in the virtual organization, but assigns jobs to the local schedulers of the various institutes; local schedulers will match jobs to resources. An exemplary system of this model is the Community Scheduling Framework (CSF) [17].
- Distributed model: multiple local schedulers exist in a VO; each local scheduler has a companion meta-scheduling functional entity; local schedulers can submit jobs to each others through their respective meta-scheduling functional entities. Exemplary systems of this model include IBM Tivoli Workload Scheduler Loadleveler [18] and Gridway [19].

All three models have their respective advantages and disadvantages and are suitable in different deployment environments. The centralized model is relatively simpler than other models. However, the meta-scheduler can become a bottleneck for a VO that has a very large number of resources. The meta-scheduler can also be a potential single point of failure. The hierarchical model is a more scalable scheme than the central model, but the meta-scheduler has less control of the scheduling decisions and can still be a single point of failure. The distributed model is the most complex of the three models and does not present bottleneck and single point of failure exposures.

For LA Grid, our meta-scheduling design is a mix of the hierarchical and distributed models, as shown in Figure 3.

Our grid model consists of multiple domains. Each domain has its domain meta-scheduler and consists of a collection of local dispatchers, local schedulers or even other meta-schedulers. A domain can be viewed as the meta-scheduling functional

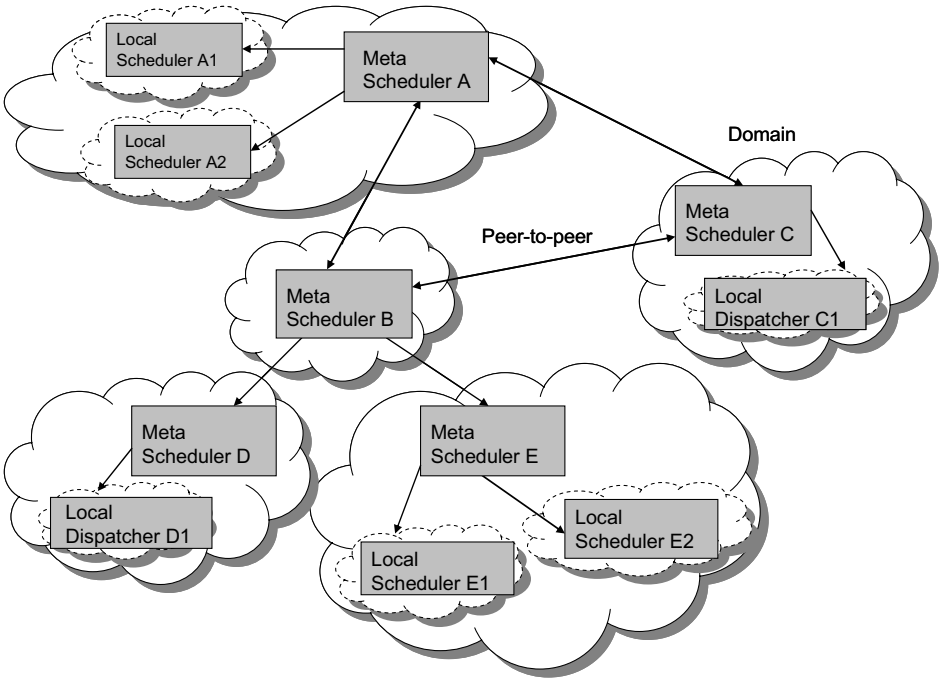


Figure 3. LA Grid meta-scheduling architecture

entity of an institution. This aspect of our model intends to reflect the reality of many organizations having multiple local schedulers for different lines of business or for various levels of services. The domain meta-scheduler supports the encapsulation of resources and scheduling details within each organization and only its aggregate scheduling capability and capacity would be exposed to the partner organizations of the grid. Then, the grid can be viewed as a sphere with collaborative partners. The peer to peer relationship between domain meta-schedulers is dynamically established upon the agreement between peers. Users of a domain would interact with that specific domain meta-scheduler to access resources of collaborative partners.

The following example illustrates the use of our meta-scheduling model. A bioinformatics data service provider differentiates its services to paying and trial customers. Paying customers are users of a consumer organization that pays for the bioinformatics services. They typically get an expected quality of service as well as access to the full set of databases. Multiple sites are set up to support paying customers with guaranteed service quality and availability. In contrast, trial customers are users without any organization association or users from a non-paying organization. They typically get service on a best effort basis on a single site with access to only sample databases. Using our meta-scheduling model, this bioinformatics provider would set up a meta-scheduling domain with a single local scheduler for trial customers. For the paying customers, the provider would either set up a sub-domain meta-scheduler with multiple local schedulers for each site, or include all the local schedulers in the same domain with a single meta-scheduler as for trial customers. For a consumer

organization, the bioinformatics data service is made available to its users (e.g. by including the data service in the service registry). Depending on the demands of its users and budgetary constraints, the organization would establish either a paying or a non-paying peer-to-peer relationship with the bioinformatics provider. For the users in the consumer organization, their application logic would not be affected by the paying status of the organization.

The Open Grid Forum [20] is leading the effort of defining a standard description language for job submission, called Job Submission Description Language (JSDL) [21]. This language allows specification of job characteristics as well as resources required for the job execution in a grid infrastructure. The adoption of JSDL is a good first step towards achieving collaboration across virtual organizations from a job execution standpoint. However, there is a strong need for a standard interface for expressing meta-scheduler to meta-scheduler interactions and meta-scheduler to local scheduler interactions to realize collaborative job execution. One of our project objectives is to experiment with the necessary interfaces to support interactions between domain meta-schedulers and their associated local schedulers. We categorize these interfaces into three sets:

- Meta-scheduler connection API: used to establish and terminate the connection between domain meta-schedulers, either through a peer-to-peer relationship or an up-stream relationship in a hierarchy. Once the connection is established, heart beats are exchanged to guarantee the healthy state of the connection.
- Resource exchange API: used to exchange the scheduling capability and capacity of the domain controlled by the meta-scheduler; the exchanged information can be a complete or incremental set of data.
- Job management API: used to submit, re-route and monitor job executions across the network of (meta-)schedulers.

A domain meta-scheduler supports these three APIs and implements the necessary functions, as illustrated in Figure 4. It is composed of three functional modules: the resource management module, the scheduling module, and the job management module. The resource management module is responsible for resource discovery, resource monitoring and resource information storage. The resource information storage can be either a persistent storage device or a cache device. The scheduling module is responsible for locating suitable resources or a suitable scheduler for each job request. The job management module manages the lifecycle of the job, including the reception of the job request, its routing or dispatching to the matched resources or scheduler, and the monitoring of the job status.

Our current implementation of the meta-scheduling APIs is using grid web services in order to more easily accommodate existing meta-schedulers and integrate them as collaborative job execution partners using the Globus Toolkit. We will verify the possibility for the same set of APIs to be recursively useful regardless of the relationship between meta-schedulers. Our experimentation platform consists of three collaborative meta-scheduling partners: the first one is based on IBM's Tivoli Dynamic Workload Broker [22]; the second one is based on the Barcelona Supercomputing Center's eNANOS broker, and the third one is based on Gridway or CSF.

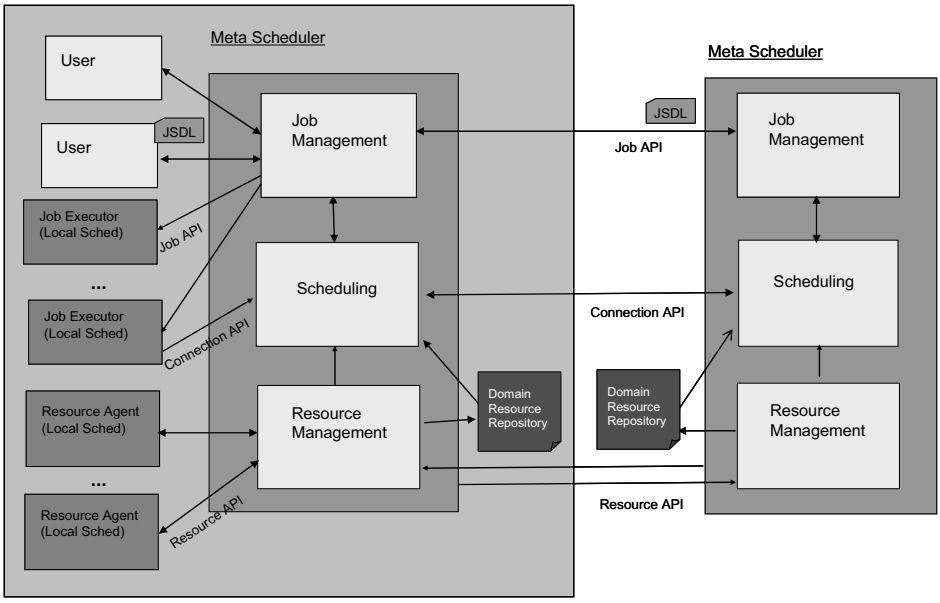


Figure 4. LA Grid domain meta-scheduling

We use the bioinformatics and hurricane mitigation scenarios depicted in Sections five and six as application test cases, as they both exhibit data and compute intensive workloads.

3. Job Flows

For many years, workflow[1] technology has been used in orchestrating multiple tasks in business processes. Only recently, scientific communities became very active in exploring workflow technology for orchestrating the execution of composite jobs that consist of multiple steps. The use of job flow would potentially provide richer expressiveness and flexibility for users to instruct the job management system on how to schedule and execute their jobs. In this part of our research, we explore issues related to job flow in grid environments, including job flow modeling, transparent workflow adaptation, and data dependencies in job flows.

3.1. Background

Job flow management can be achieved through service orchestration or choreography [23]. In service orchestration, job flow management is achieved through a central application. This application (usually an executable workflow) models the interaction between the partner services, so that they collectively accomplish a coarse grain task. The application is aware of the interfaces of the partner services and controls their execution order and message exchanges. In service choreography, job flow management is achieved through a distributed approach, where partner services are

[1] In this document, we use the terms *job flow* and *workflow* interchangeably.

aware of each other and each service knows of its participation in the message exchanges of the interaction. Figure 5 illustrates the difference between orchestration and choreography.

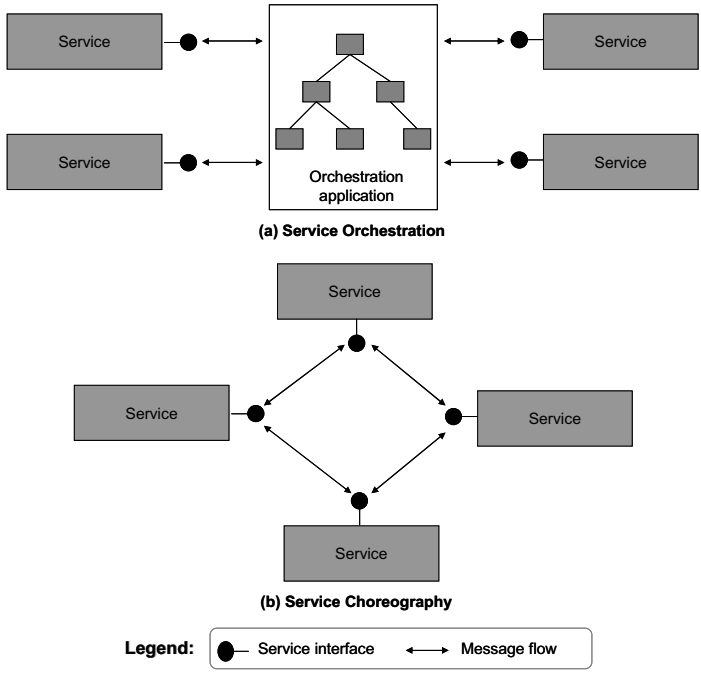


Figure 5. Orchestration and choreography

3.2. Overview

We have adopted a two-level approach to job flow management in grid environments that employs *service orchestration* at the upper level to control coarse-grain job submissions and *service choreography* at the lower level to control the interactions among executing jobs. To create high-level service orchestration for job submissions, we use the Business Process Execution Language (BPEL) [24], which has become the leading standard for web service orchestration. Web services can be integrated, using some XML-grammar, to create a higher-level application (business process). The XML-grammar that defines a BPEL process is interpreted and executed by an orchestration engine which exposes the process itself as a web service. BPEL provides many constructs for the management of process activities, including loops, conditional branching, fault handling and event handling (such as timeout). Additionally, it allows for activities to execute sequentially or in parallel. For the lower level choreography, we use JSDL [20] to describe the requirements of jobs for submission to the grid.

In addition jobs also need one or more input data items for their execution and can produce multiple outputs as well. There may be multiple copies/replicas of these data items in the system. For replica management in grids we use the Replica Location Service (RLS) available in the Globus Toolkit. RLS maintains and provides access to mapping information from logical names of data items to target names. These target names may represent the physical locations of the data items or map to other entries in

the RLS providing a second level of logical naming for the data items. The use of logical to physical name abstraction provides users with the option of specifying logical names of data items in their job descriptions, rather than the actual physical locations. Also, in distributed grids, it is very often desirable to maintain multiple copies of data items so that job executions can be optimized at more than one possible location.

Figure 6 provides an architectural diagram of our job flow management framework and shows how it interacts with meta-schedulers. First, the Grid Job Flow Application Modeling and Tooling captures the job control flow in an abstract BPEL workflow and the data dependencies as embedded JSDL scripts within the BPEL workflow. These documents are then interpreted to extract the job to job, job to data, and job to resource dependencies to form a directed graph. Next, this graph becomes more concrete by mapping the jobs to scheduling domains considering the resources they need, the data they require, and the other jobs they are dependent on. Then, additional steps for data transfer and Replica Location Service (RLS) registration are appended to the BPEL workflow. At this point the abstract BPEL workflow is concretized by binding the jobs to specific resource domains.

Any good job flow management system must adequately address the issue of fault-tolerance on behalf of the job flow. BPEL has constructs for detecting (and generating) fault messages, as well as constructs for specifying event-driven compensation activities. Compensation activities serve to undo some business logic that occurred prior to the event. However, grid environments call for more robust fault handling than is available in BPEL. To make job flows resilient to failure, in this step, we use a previously developed framework, called TRAP/BPEL [25], which adds autonomic behavior into existing BPEL processes automatically and transparently.

Unlike other approaches, TRAP/BPEL does not require any manual modifications to the original code of the BPEL process and there is no need to extend the BPEL language, nor the BPEL engine. Within the TRAP/BPEL framework, a BPEL workflow is made adaptive by first running it through an adaptation generator. The generator generates the adapt-ready BPEL process by inserting "hooks" at specified points in the workflow. These hooks redirect invocations through a proxy that provides adaptive behavior by shielding the workflow from failure and applying recovery mechanisms that are specified in a recovery policy [25].

During adaptation, specific jobs which require monitoring are identified and for each job, an adequate failure handling technique is specified in a recovery policy. The recovery policy is modeled in an XML document that is not part of the job flow definition. Failure handling techniques may include check-pointing, or finding an alternative (substitute) resource or service upon which to submit a job. Invocations for monitored jobs are replaced with invocations to a generic proxy. Messages for those jobs are therefore redirected through this proxy. The proxy using some failure detection mechanism (e.g. polling, event notification) monitors the individual jobs and enforces the recovery policy. The proxy in this case is a job submission and monitoring service and forwards jobs to the schedulers. There is only one generic proxy per job flow engine, although there may be several instances of this proxy performing recovery on behalf of the workflows executing on that engine.

Finally, the adapted concrete BPEL process with embedded JSDL scripts is executed on a BPEL engine to orchestrate the job submission through the use of some job submission web services. As described in Section two, meta-schedulers can engage in peer-to-peer choreography in order to decide on the specifics of how the actual job

execution and data transfers occur (e.g. the exact machine on which the job is to be executed, the exact data transfer protocol that is to be used).

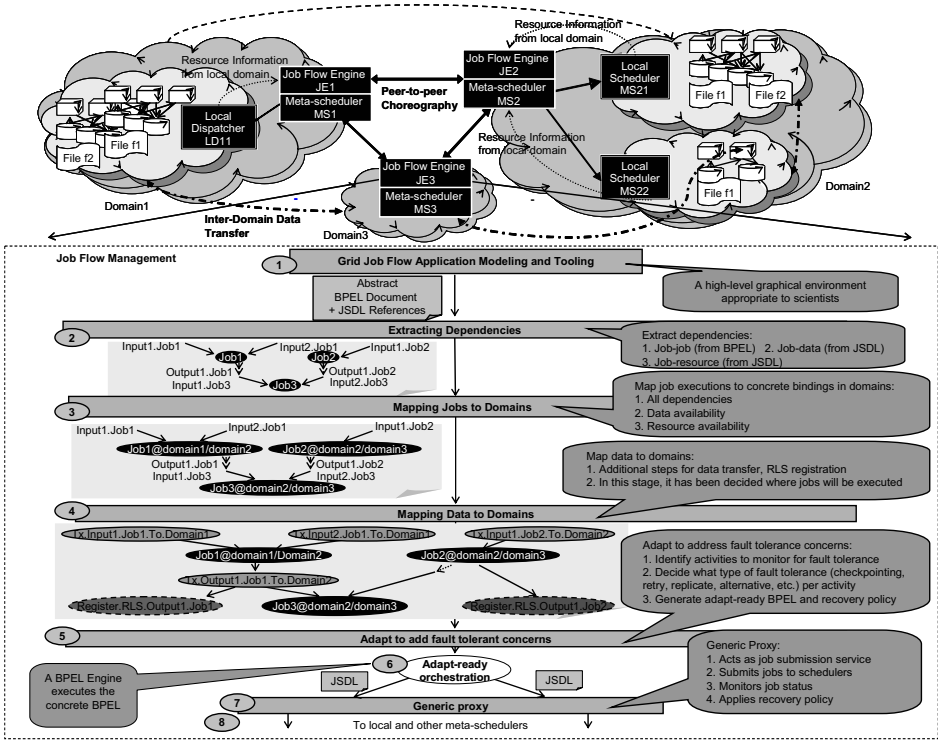


Figure 6. Job flow management framework interaction with meta-scheduling

3.3. Related Work

There are many job flow languages used in the scientific communities including DAGMan of the University of Wisconsin [26], SCUFL of the e-Science group [14], YAWL of Queenland University Technology [27]. In the enterprise domain, the most popular language for business processes is WS-BPEL [24]. WS-BPEL has become the de-facto standard for workflow technology in enterprise systems. To help foster interactions between job flow systems in the grid there is clearly a similar need for standardization. For this reason, in recent times, the scientific community has also started exploring the possibility of using the WS-BPEL standard for job flows. We selected WS-BPEL as our workflow language for our project. Software offers that support WS-BPEL include IBM's WebSphere Process Server [28] and ActiveBPEL [29].

There are many projects and studies of job flows in grid environments. The Pegasus [30] project at the University of Southern California addresses the planning and resource allocation of job flow in grid environments. VDS of GriPhyN [31] supports the virtual data specification language used in conjunction with the DAGMan job flow language. Unlike other studies, our project explores the combined use of WS-BPEL and JSDL along with data mapping.

Our architecture also supports the specification of flexible, user-defined failure handling mechanisms while supporting separation of concerns. By adapting the concrete BPEL workflow, failure handling mechanisms are defined in a manner that does not tangle the code for fault-tolerance with the business logic of the application. Some other Grid workflow systems (Karajan [32], Kepler [33] and Grid-WFS [34]) allow for user-defined failure handling, however, Karajan and Kepler do not support the separation of concerns. Grid-WFS claims to support the separation of concerns but fault handling strategies are specified along with the task in the same workflow definition. This approach does therefore entangle the code for failure handling with that of the business logic. Further, our approach does not require a purpose-built workflow engine, as do the other systems, since we utilize standard BPEL constructs.

For more general information on workflows, the reader is invited to refer to http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf, which is a good survey paper on workflow taxonomy and workflow projects.

4. Data Integration

Data integration involves combining data residing at different sources and providing the user with a unified view of such data. Data integration is a traditional problem that exists in numerous applications, for example, integrating the databases for two companies that are being merged, or combining research results from different bioinformatics repositories. Data integration frequently occurs as the volume of data and the need to share it increase. Data integration has been the focus of extensive work and numerous open research problems remain to be solved. In this study, we present a system architecture which aims to define essential components for developing a domain-specific, modular, and decentralized data integration system in a grid environment. The proposed architecture incorporates a series of grid-oriented services (e.g. coordinator, semantic catalog, repository, synchronization) that address the distributed nature and autonomy of the data sources. The architecture also incorporates a series of data-oriented services (e.g. schema mapping generation, query generation, query rewriting) that facilitate the actual integration of data. Nodes in the grid environment can provide data, integration services, or a combination of both. Through the proposed system, users can contribute new data, define relationships among existing data sources and schemas, relate data to domain-specific concepts, or even construct new schemas that can be reused by others.

4.1. Architecture and Services

Our architecture supports distributed storage and manipulation of data and adapts to dynamic addition and removal of nodes. For every session, an application server connects users to nodes and designates a particular node as the “master”. Any node is capable of performing the “master” role. The master node distributes the required tasks among many other nodes in the grid and is also responsible for coordinating, collecting, and merging results from “slave” nodes.

Figure 7 depicts the LA Grid data integration architecture. Each LA Grid node contains seven components/services, namely: the *coordinator service*, the *semantic catalog*, the *repository service*, a *data repository*, the *synchronization service*, the *Data Grid Management System (DGMS)* [35], and the *data services suite*. At this time the

data services suite is completely implemented and other services are at different stages of development. Implementation is done in Java and nodes are currently deployed in a grid architecture based on the Globus Toolkit [5].

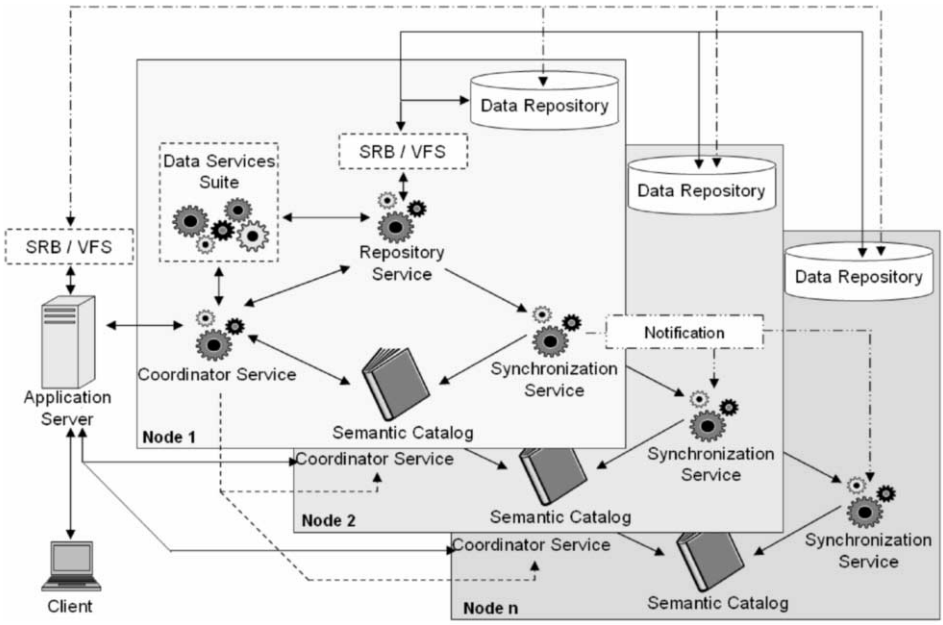


Figure 7. Data integration architecture

Coordinator Service: This service is the front-end interaction point for data-driven operations. It has access to a locally held semantic data catalog. The local coordinator service communicates with remote coordinator services for coordinating operations across the system. Query distribution/data materialization decisions are also handled by this service based on quality of service, load balancing, and optimization requirements. Coordinator services are key elements for automated workflow construction because they are responsible for decisions involving forwarding, splitting, or directly handling a request. Upon receiving a request, the coordinator needs to access the semantic catalog to retrieve information about data needed/involved in the request. Based on metadata and information about registered nodes and deployed services, the coordinator makes a decision about handling the request, i.e., forwarding, splitting, or directly handling it. The coordinator maintains a set of rules to help make such decisions. The efficiency and accuracy in defining such rules is crucial for correct and efficient system performance. While a system could perform its function using a limited set of rules, its performance could be enhanced by adding and tuning rules. For example, a simple rule is to forward a request to the first node that has the required services deployed. However, a better rule is to incorporate the location of data. From our experience, fine tuning rules may result in complex but efficient workflows of services.

Semantic Catalog: The semantic catalog contains the physical locations of data components as well as domain-specific semantic descriptions. For example, the semantic catalog of each schema may store its name, the location of its definition,

semantic mappings from the schema elements to biomedical domain-specific concepts, and pointers to known instances of the schema. Additionally, we may store schema mappings that directly relate pairs of schemas (created by the schema mapping creation service – cf. detailed description below). The semantic mappings between schema elements and domain-specific concepts are currently simple correspondences from schema elements to terms in a conceptual model, similar to an ontology. Users provide ontologies and mappings applicable to their problem domain. This kind of semantic mapping is an active research area [36] that can enhance schema mapping/matching operations (for example, if two or more schema elements can be mapped to the same semantic term in the ontology, a potential match is indicated). In the current implementation, UMLSKS [37] is used as the domain-specific mappings knowledge source. The latest version of the semantic catalog is built as a dynamically evolving OWL (Web Ontology Language) [38] resource. It captures relations between schema definitions, instances, mappings and their domain. Those relations are represented as RDF (Resource Description Framework) [39] statements that can be manually asserted by system users or can be automatically evaluated using a set of utility services. Using this design, the semantic catalog is able to answer queries like: Is there a mapping from schema A to schema B?; Find all mappings that use schema C as a source; Does a mapping have an inverse in the repository?; etc. Pellet [40] - an open source, OWL Description Logic reasoner in Java - is used for reasoning and augmenting information in the catalog while SPARQL [41] is used as the query language.

Repository Service & Data Repository: This service is responsible for storing and extracting all raw data (via the DGMS or VFS - discussed below). It also notifies the synchronization service about new changes in the repository. Currently, this service is implemented on top of the Apache Commons Virtual File System [42]. VFS provides APIs for accessing different file systems and presents a unified view of files from different sources (e.g., local disk, remote ftp, or http servers). In the current system implementation, only a pure XML data repository is supported. Other data representations can be supported only if the data can be exported to XML.

Synchronization Service: The synchronization service keeps the semantic catalog entries synchronized among nodes. When a node is added to the grid, the node has the option of subscribing to various topics. Whenever a change affecting a given topic occurs, the nodes subscribed to that topic receive notifications of the update. This service is implemented on top of the WSRF notification mechanism provided by the Globus Toolkit.

Data Grid Management System: Through multiple abstractions, the DGMS [43] provides a logical namespace that hides the complexity of distributed data and heterogeneous resources. The Storage Resource Broker (SRB) [44] is a tool for managing distributed storage resources. Files in the SRB are referenced by logical file handles that do not require the actual physical locations of the files. A Metadata Catalog, MCAT, maintains maps of logical handles to physical file locations. The proposed semantic catalog and data services can be seen as augmentations on top of DGMS in order to facilitate finer grain semantic integration at the data level. The current system implementation uses the Apache VFS.

Data Services Suite: This component provides a number of web services that allow the creation of schema mappings and operations over those schema mappings. We have

selected Clio's [45] schema mapping components, and wrapped them as web services. The suite provides the following data services:

- ***Schema Mapping Creation:*** Given a source schema, a target schema, and a set of "correspondences" between source and target schema elements, this service creates a "mapping" from the source schema to the target schema. This mapping consists of a set of declarative constraints that dictate what the target instance should be, given a source instance. The mapping creation algorithm takes into account the schema constraints (e.g., foreign key constraints, type constraints) as well as correspondences [46].
- ***Query Generation:*** Given a mapping (produced by the Schema Mapping Creation service), this service produces an XQuery, an XSLT, or an SQL/XML query that implements the transformation implied by the mapping [45]. The query and the association between the query and the mapping used to produce the query are stored in the semantic catalog (for future reuse).
- ***Query Execution:*** For convenience, we also have a service that executes the queries generated by the previous service. Given a query script and a set of input XML documents (e.g. instances of the source XML schema), the service executes the query and returns the resulting XML document.
- ***XML Transformation:*** This service allows the direct and scalable execution of the mapping, as opposed to simply executing the query that implements it. Based on the technology detailed in [47], this service takes as input a mapping and the source XML instances and returns the target XML instance that is implied by the mapping. As opposed to the query generation/execution services, this service neither produces nor executes a query; rather, it uses a Java-based engine to optimally execute the mapping.
- ***Query Rewrite:*** An interesting application of mappings is the ability to rewrite target-side queries into queries that work on the source-side. This is useful, for example, if the target-side schemas are virtual and actual data resides on the source side. We use the query rewriting techniques detailed in [48] to implement this service. Given a schema mapping and an XQuery over a target schema instance, this service returns a rewritten XQuery over the source schemas in the mapping.
- ***Schema Integration:*** Given a number of mappings between several schemas, this service attempts to create an "integrated" schema that captures the unified concepts of the schemas that are related by the mapping [49].

4.2. Related Work

A number of data integration systems have been proposed to address the problem of large-scale data sharing (e.g. [50, 51, 52]; and the survey by Halevy [53]). These systems support rich queries over large numbers of autonomous, heterogeneous data sources by making use of semantic relationships between the different source schemas and a mediated schema, which is designed globally. However, the mediated schema becomes a problem since it may be hard to come up with a single mediated schema that everyone agrees on. Moreover, all access (querying) is done against the mediated schema (a single point). Furthermore, this architecture is not robust with respect to the changes in the source schemas. As a result, data integration systems based on mediated schemas are limited in supporting large-scale distributed and autonomous data sharing. Peer Data Management Systems (PDMS), e.g., Piazza [54], have been proposed to address the aforementioned problems and to offer an extensible and decentralized data

sharing system. The study presented in this section can be viewed as an effort to present and discuss the design, components, and services required to realize a PDMS in a grid environment. Our system requirements are, in principle, no different from these peer data management systems. Compared to Piazza's approach, our intended applications imply smaller numbers of data sources. However, the sources have complex schemas and may contain overlapping and potentially conflicting and dynamically changing data. The proposed system emphasizes the use of tools and services that facilitate mappings among schemas and generate the queries that are needed to access and integrate the data.

5. Application to Bioinformatics

A problem facing many bioinformatics researchers today is the aggregation and analysis of vast amounts of data produced by large scale projects such as the Human Genome Project. This is further complicated by the fact that data is distributed among heterogeneous sources. As of September 2006, the Gene Expression Omnibus (GEO) repository at the National Center for Biotechnology Information (NCBI) holds over 3.2 billion individual measurements. Moreover the repository is growing at a rapid rate [55]. The amount of data, the rate of its growth, and the heterogeneity of data sources present real problems, hindering advancements in bioinformatics [56].

In this section, the focus is on data integration problems in bioinformatics. However, a typical bioinformatics research activity involves both computational and data driven aspects. Data driven tasks involve techniques to extract data from multiple sources. Computational tasks involve processing data after extraction for pattern matching, alignment, and clustering.

Pharmacogenomics is a branch of bioinformatics dealing with the influence of genetic variation on drug response in patients. Approaches investigating such influences promise the advent of "personalized medicine", in which drugs and drug combinations are optimized for each individual's unique genetic makeup. To make "personalized medicine" decisions, information from multiple heterogeneous data sources needs to be incorporated; for example OMIM, dbSNP and dbGaP from NCBI [57], Haplotype data from the HapMap project [58], Human Gene Mutation and TRANSFAC databases from BioBase [59] in addition to PHARMKGB [60]. Figure 8 shows a related example that aims to understand rates of gene expressions in different tissues and correlate these expression profiles with active transcription factors and their binding sites. The data required for this study is distributed among various sources (e.g. UCSC Genome Browser [61], GNF SymAtlas [62] and TRANSFAC [63]). The figure shows how Clio mapping technology can be used to provide a high-level definition for mappings between such sources and a target schema. In particular, a graphical user interface allows the identification of correspondences that relate schema elements.

Our sample scenario involves three collaborating groups of scientists. Assume the groups are associated with the three data sources in Figure 8 and located in the USA, Spain, and Mexico, respectively. The USA group is conducting experiments related to identifying known genes and their chromosomal positions. The team from Spain is doing experiments on gene expression levels in different tissues, while the team from Mexico is concerned with identifying transcription factors binding sites for different genes and the associated transcription factors. Furthermore, assume there is a fourth team in the UK that will do the analysis of the collected data. Their role is to collect

and interpret data from the different teams and to discover new knowledge from the experiments conducted in the study.

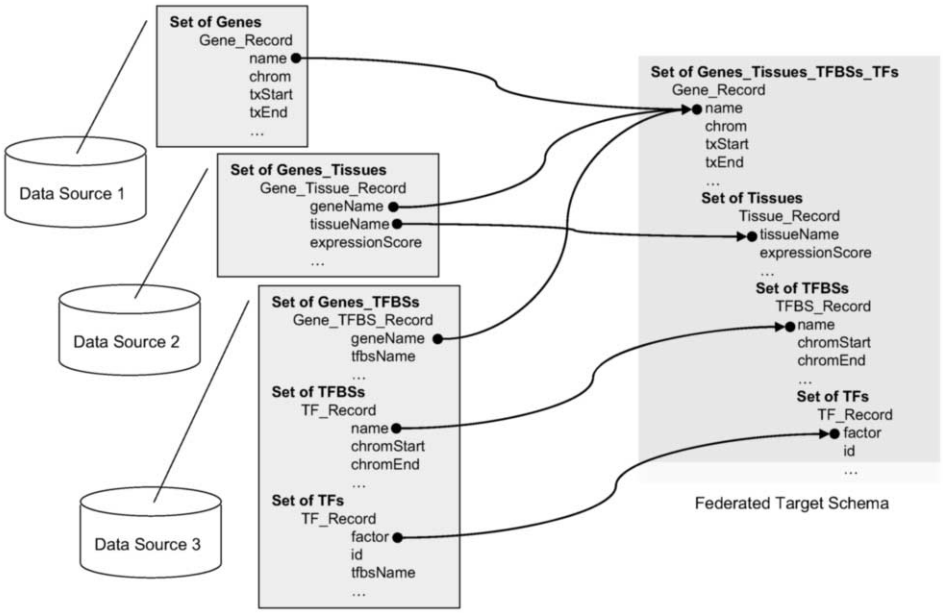


Figure 8. A high-level mapping definition using Clio

Assume that the teams in the USA, Spain and Mexico have uploaded their data to their associated data sources. Now the UK analysis team can start interpretation and analysis of the data. However, they are facing the problem of merging and integrating these three data sources. To efficiently analyze the data, they would like to organize it according to a specific structure. Therefore, the UK team constructs a new schema that captures the required data organization (the target schema in Figure 8) and creates the required domain-specific mappings. Then, they connect to their node and download (via Java Web Start) an application that allows the construction of Clio-based mappings. Source and target schemas are loaded into the tool which shows their structure as a tree of schema elements (very similar to how they are presented in Figure 8). Value mappings are entered by drawing lines from source schema elements to target schema elements. The output from this process is a value mapping, which is sent to the schema mapping service that evaluates the mapping specification and passes it to the repository service for storage. The synchronization service updates the local semantic catalog and notifies remote nodes about the new mapping.

Different query processing scenarios could arise based on the location of data (local vs. remote), and whether the query is against a materialized version of the data or not. If the data is not materialized then either a materialization or a query distribution decision could be made by the coordinator service. Criteria for such decisions can be based on the frequency of the queries against data sources.

For instance, the UK team is trying to answer the following query using the federated target schema:

"Find a list of ***gene names*** *and* ***their chromosomal locations*** *that have an* ***expression level > e*** *in* ***both heart and liver*** *and are* ***regulated by the same set of transcription factors.****"*

The above query is written against the target schema. However, it is assumed that the UK node does not have any data associated with this federated schema; all data resides at the other nodes (a *global-local-as-view* –GLAV– scenario [64]). The query is rewritten by the Query Rewrite service into a new query, formulated in terms of the source schemas (at the USA, Spain, and Mexico sites). When executed, the rewritten query retrieves the three source documents and then locally (on the UK node) joins and filters the data, and finally produces an instance of the target schema. Another alternative we are exploring is to further decompose the rewritten query into maximal sub-queries that are sent to the sources. For example we could send a join query to data source 3 (Mexico) and only get the relevant data back. In another scenario, one or more source schemas may also be the output of previous schema mapping operations. In such cases, a nested query rewrite with further decomposition is needed.

6. Application to Hurricane Mitigation

6.1. A Possible Hurricane Mitigation Scenario

A tropical depression in the Caribbean Sea quickly strengthens in the warm waters as it drifts westward into the Gulf of Mexico. It is tracked and modeled by the National Hurricane Center as it becomes a tropical storm and then a category one hurricane by day three. Once in the central Gulf of Mexico the storm intensifies into a category three hurricane by the end of day four. A similar storm is shown in Figure 9.

Hurricane track models begin to indicate that the storm will continue to intensify and re-curve northeast and then east toward the western Florida coast. The NOAA National Hurricane Center model forecasts begin to predict landfall along the central western Florida coast by day six.

Synoptic scale numerical weather prediction (NWP) models capture the general storm circulation and general movement, but do not predict intensity changes well [65, 66]. Results of these large scale models, in turn, are used along with other high resolution data as input to regional and mesoscale models that run ensembles across a computing grid infrastructure of thousands of nodes. These ensemble models do more than determining high resolution hurricane impact; they also provide information about the uncertainty of the hurricanes track and intensity forecast. Between 48 and 72 hours prior to landfall these ensembles allow risk management for the event, as they also include information about the sensitivity of the forecast to both data and physics uncertainty.

6.2. Our approach

The Weather Research and Forecasting (WRF) model is the state-of-the-art mesoscale numerical weather prediction system, serving both operational forecasting and

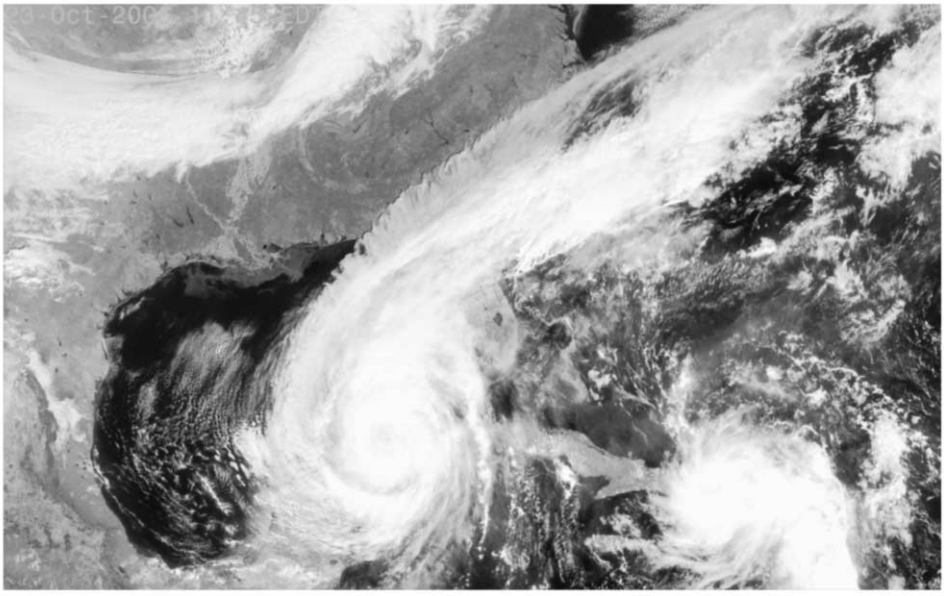


Figure 9. Hurricane Wilma in the Gulf of Mexico - October 23, 2005

atmospheric research needs [67]. The WRF model Version 2.1.2 software distribution comprises about 360000 lines of source code. This code is highly modular and is greatly optimized to run on several heterogeneous cluster computing facilities using MPI [1] for inter-node communications and OpenMP [68] for intra-node communications among the processors. In this part of our research, we address two problems with the current version of WRF.

First, to mitigate the impact of hurricane landfalls, we need to provide even more accurate and timely information to enable effective planning [69, 70, 71, 72]. Pushing the limits of WRF today, there is a increasing need for fine-grain simulation in smaller regions (e.g. zip-code level hurricane simulation). Considering the limited resources available—that each interested organization may have—would leave us with no choice than to scale out from single-managed cluster computing to grid computing that can span over many organizations with different needs and administrative domains. Such grid-enabled WRF code can employ the resources available in several organizations to contribute toward solving fine-grain hurricane simulations. As the WRF code does not provide any inherent support for grid computing and as we do not want to stray away from the mainstream revisions of the WRF code, we are employing our Transparent Grid Enablement approach (mentioned in Section one) to enable execution of WRF on grid computing environments in a transparent manner to the original WRF code (no manual modification to the WRF code).

Second, to contribute to future WRF model development, meteorologists are required to develop new dynamics and physics model packages in languages such as FORTRAN and C as well as to understand the architecture of the underlying

computing platforms to optimize code – very challenging skills. In this part of our research, we address this problem by investigating on a high-level modeling platform, a web-based interface called the Grid WRF portal. The portal allows meteorologists to use WRF and to develop new meteorology model packages adaptable to different grid computing environments. Our preliminary work shows that a specialized visual modeling interface supported by a workflow language (e.g. BPEL) may be a good choice. We continue to address challenging questions such as the expressive power of workflow languages [73, 74, 75, 76], and the efficiency of the generated WRF code [77]. This high-level graphical user interface is backed up with the results of our other research mentioned in Sections two and three, where we have designed and partially developed a job flow management system and a meta-scheduler that can adapt to the dynamic changes of a grid computing environment.

Beyond grid enablement of hurricane modeling across multiple geographic and temporal scales for implementation of real time ensemble simulations, there is the potential for creating a suite of products both distinct and integrated which couple multiple types and scales of model simulations (atmospheric, ocean wave, storm surge, hydrological, socio-economic models, etc.). This capability coupled with user- and application-centric visualizations gives rise to a new class of data analytics for decision support in a proactive sense that has heretofore been unavailable.

Ocean wave and storm surge models are also run on the grid and are used to better predict impacts on coastal infrastructure, shipping and oil drilling interests in the eastern Gulf of Mexico and along the shoreline as the hurricane continues its eastward track. Mean and ensemble members of the atmospheric, ocean wave, storm surge and hydrological models are coupled to create predictions on storm movement, intensity, near shore wave heights, storm surge, and flooding forecasts that include both optimistic and pessimistic bounds on the likely outcomes. The various model forecasts are concurrently produced and visualized with grid resources to create customized decision support guidance for emergency management, utility, transportation, debris removal and other interests as the storm makes landfall along the western Florida coast.

National authorities do not currently have the required computational power nor the bandwidth to produce such high resolution (cloudscale) ensembles and deliver them to emergency management, businesses, and the public. Within the grid environment, smaller domain storm surge results and visualization of complex rainfall, wind fields and waves can be generated for distinct localities in high detail. Mesoscale and cloudscale model ensemble forecasts continue to run in real time utilizing grid resources to enable timely generation of decision support products. Information, data and analytics are produced in multiple forms, which are then integrated and delivered to support planning, response and remediation as the storm moves eastward across Florida and exits along the northeastern coast. These analytics allow improved estimates of damage to electrical grids and transportation infrastructure while supporting efforts to plan and mobilize storm recovery.

Grid enablement for the generation of new and unique products, guidance and analytics in support of mission critical decision making in a transparent manner while utilizing a dynamic, heterogeneous and geographically disparate set of computing resources is one of the strategic goals of the LA Grid hurricane mitigation project. It is a necessary and evolutionary step in the convergence of science and technology for societal benefit.

7. Future Plans

Our initial work demonstrated in previous sections is part of a novel approach to grid application development we call Transparent Grid Environment (TGE), whose goal is to allow domain experts to effectively express the logic and software artifacts of domain applications while hiding the details of the grid architecture, software, and hardware stack. This TGE paradigm will serve as the foundation for the study of application development methodologies, platforms, and tools that will significantly ease grid-enabled application development (hence broadening grid utilization) and make applications more portable and adaptable to future changes of grid technologies. We believe that grids utilizing the TGE paradigm will be agile, flexible, and capable of serving a broad set of scientific and business communities.

Our approach is characterized as application-driven (hence "top-down") by basing and focusing our investigation on (1) supporting grid-enablement for a few carefully chosen critical application domains, e.g. hurricane mitigation and bioinformatics, and (2) developing common methodologies, services and tools for deploying grid-enabled applications in these domains. In our approach, we factor out common services that can be reused across domains. This will ensure that our tools have broad significance and utility to a range of applications, thus avoiding the tendency for tools to be too generic to be effective. Our future plans entail investigation of the following key challenges:

1. *High-level Visual Interactive Development Environment (IDE)*: What is the appropriate IDE for domain experts to easily specify the logic of their applications? Do IDEs targeted for different domains have many common properties? Is it possible to develop a common IDE that supports multiple domains? Are the workbenches being developed and used by domain experts today the right solution?
2. *Automated Code Generation and Software/Hardware Reuse*: How can we automate the generation of executable code from a high-level specification provided by a domain expert? How can we reuse the existing software and hardware components and map abstract specifications to concrete resources in order to execute the application?
3. *Hiding the Heterogeneity of Grid Architectures*: How do we hide the details of heterogeneous grid architectures and resources and provide a virtualized interface for application development while addressing efficient resource utilization?

Under the LA Grid initiative, we have established a globally-integrated research and education program to respond to the above challenges and to realize a TGE. We have taken a divide-and-conquer approach that allows us to tackle the above challenges in parallel. To conduct our investigations we have identified nationally-important domains of *Grid Applications,* as discussed in Sections five and six. We have established a number of *Grid Integration* projects that enable aggregation and discovery of data, visualization of data, and weaving of high-level grid enablement services into the logic of domain-specific applications, as described in Section four. Finally, we have established several *Grid Enablement* projects that provide a layer of abstraction on top of heterogeneous Grid architectures by offering high-level services, as presented in Sections one, two, and three.

As our research progresses, the technologies and their associated tools developed in these projects will form grid application enabling platforms at different levels of abstraction. First, the projects in the Grid Applications Layer will identify the

requirements and provide methodologies, frameworks, and modeling tools that enable domain experts to model, design, and code their applications with minimal attachment to the underlying grid. In designing and developing these "application interfaces," common high-level, application-oriented functions, components, and tools are extracted and packaged into the Grid Integration Layer, which provides services that can be reused by other applications in the targeted domains. The projects in the Grid Enablement Layer will provide system-level services and tools to support efficient and transparent management and utilization of heterogeneous grid resources through uniform virtualized interfaces. Such services will make applications even more adaptable to changes of the underlying infrastructure.

The LA Grid initiative aims to simplify the manner by which scientific and business domain experts develop, use, and maintain software applications over distributed computing resources. Using our TGE approach we will create innovative tools which promote the reuse of commonalities across domains resulting in flexible and cost effective grid implementations which allow experts in diverse domains to easily code their application logic using an integrated development process.

References

[1] http://www-unix.mcs.anl.gov/mpi/.
[2] http://www.globus.org/ogsa/.
[3] http://www.globus.org/wsrf/.
[4] http://www-unix.mcs.anl.gov/mpi/mpich2/.
[5] http://www.globus.org/toolkit/.
[6] http://www.unicore.org/.
[7] http://www.cs.wisc.edu/condor/.
[8] R.M. Badia, R. Sirvent, J. Labarta, J.M. Perez, Programming the GRID: An Imperative Language Based Approach, Book chapter in *Engineering the Grid*, Section 4, Chapter 12 , January 2006.
[9] S.M. Sadjadi, P.K. McKinley, B.H.C. Cheng, and R.E.K. Stirewalt, TRAP/J: Transparent generation of adaptable Java programs, *Proc. International Symposium on Distributed Objects and Applications (DOA'04)*, Agia Napa, Cyprus, October 2004.
[10] R.V. van Nieuwpoort, J. Maassen, T. Kielmann, and H.E. Bal, Satin: Simple and efficient Java-based grid programming, *Scalable Computing: Practice and Experience* **6(3)**, 19-32, September 2005.
[11] S. Gorlatch and J. Dunnweber, From Grid Middleware to Grid Applications: Bridging the Gap with HOCs, in *Future Generation Grids*, Springer Verlag, 2005.
[12] M. Aldinucci, M. Coppola, M. Danelutto, M. Vanneschi, and C. Zoccolo, Assist as a research framework for high-performance grid programming environments, in *Grid Computing: Software Environments and Tools*, J.C. Cunha and O.F. Rana, Eds., Springer Verlag, 2004.
[13] F. Baude, L. Baduel, D. Caromel, A. Contes, F. Huet, M. Morel, and R. Quilici, Programming, Composing, Deploying for the Grid, in *Grid Computing: Software Environments and Tools*, J.C. Cunha and O.F. Rana, Eds., Springer Verlag, January 2006.
[14] http://taverna.sourceforge.net/.

[15] V. Subramani, R. Kettimuthu, S. Srinivasan, and P. Sadayappan, Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests, *Proc. 11th IEEE International Symposium on High Performance Distributed Computing*, Edinburg, Scotland, July 24-26, 2002.
[16] I. Rodero, et al., Looking for an Evolution of Grid Scheduling: Meta-brokering, submitted to ICS 2007.
[17] http://www.globus.org/grid_software/computation/csf.php.
[18] http://www-306.ibm.com/software/tivoli/products/scheduler-loadleveler/.
[19] http://www.gridway.org/.
[20] http://www.ogf.org/.
[21] A. Anjomshoaa, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, Job Submission Description Language (JSDL) Specification, Version 1.0, November 2005, Copyright © Global Grid Forum (2003-2005).
[22] http://www-306.ibm.com/software/tivoli/products/dynamic-workload-broker/index.html.
[23] C. Peltz, Web services orchestration and choreography, *IEEE Computer*, **36(10)**, 44–52, 2003.
[24] S. Weerawarana, F. Curbera, Business process with BPEL4WS: Understanding, *http://www-128.ibm.com/developerworks/library/ws-bpelcol1/*, 2002.
[25] O. Ezenwoye and S.M. Sadjadi. TRAP/BPEL: A framework for dynamic adaptation of composite services, *Proc. International Conference on Web Information Systems and Technologies (WEBIST 2007)*, Barcelona, Spain, March 2007.
[26] http://www.cs.wisc.edu/condor/dagman.
[27] http://yawlfoundation.org/index.php.
[28] http://www-306.ibm.com/software/integration/wps/.
[29] http://www.activebpel.org/.
[30] E. Deelman, G. Singh, M.H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, G. K. Vahi, G.B. Berriman, and J. Good, Pegasus: A framework for mapping complex scientific workflows onto distributed systems, *Scientific Programming* **13** (2005), 219–237.
[31] http://vdt.cs.wisc.edu/components/vds.html.
[32] G. von Laszewski and M. Hategan, Java CoG Kit Karajan/GridAnt Workflow Guide, *Technical Report, Argonne National Laboratory*, Argonne, IL, USA, 2005.
[33] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao, Scientific Workflow Management and the KEPLER System, *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, 2005.
[34] S. Hwang and C. Kesselman, GridWorkflow: A Flexible Failure Handling Framework for the Grid, *Proc. 12th IEEE International Symposium on High Performance Distributed Computing*, Seattle, WA, June 2004.
[35] Data Grid Management System, http://www.sdsc.edu/srb/index.php/DGMS.
[36] Y. An, A. Borgida, and J. Mylopoulos, Constructing Complex Semantic Mappings between XML Data and Ontologies, *Proc. Fourth International Semantic Web Conference* (2005), 6–20.
[37] http://umlsks.nlm.nih.gov/.
[38] http://www.w3.org/TR/owl-features/.
[39] http://www.w3.org/RDF/.
[40] http://pellet.owldl.com/.

[41] http://www.w3.org/TR/rdf-sparql-query/.
[42] http://jakarta.apache.org/commons/vfs/.
[43] A. Jagatheesan, and A. Rajasekar, An introduction to data grid management systems, *Proc. SIGMOD* (2003), 683.
[44] http://www.sdsc.edu/srb/index.php/.
[45] L.M. Haas, M.A. Hernandez, H. Ho, L. Popa, and M. Roth, Clio Grows Up: From Research Prototype to Industrial Tool, *Proc. SIGMOD* (2005), 805–810.
[46] L. Popa, Y. Velegrakis, R.J. Miller, M.A. Hernandez, and R. Fagin, Translating Web Data, *Proc. VLDB* (2002), 598–609.
[47] H. Jiang, H. Ho, L. Popa, and W.S. Han, Mapping-Driven XML Transformation, *Proc. 16th International World Wide Web Conference* (2007).
[48] C. Yu and L. Popa, Constraint-Based XML Query Rewriting for Data Integration, *Proc. SIGMOD* (2004), 371–382.
[49] L. Chiticariu, P.G. Kolaitis, and L. Popa, Semi-Automatic Generation and Exploration of Schema Integration Alternatives, Manuscript under preparation (2007).
[50] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman, V. Vassalos, and J. Widom, The TSIMMIS Approach to Mediation: Data Models and Languages, *Journal of Intelligent Information Systems*, **8(2)** (1997), 117–132.
[51] O.M. Duschka and M.R. Genesereth, Answering recursive queries using views, *ODS* (1997), 109–116.
[52] I. Manolescu, D. Florescu, and D. Kossmann, Answering XML Queries on Heterogeneous Data Sources, *VLDB* (2001), 241–250.
[53] A.Y. Halevy, Answering Queries Using Views: A Survey, *VLDB* (2001) 270–294.
[54] A.Y. Halevy, Z.G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov, The Piazza Peer Data Management System, *IEEE Trans. Knowl. Data Eng.*, **16(7)** (2004), 787–798.
[55] T. Barrett, D. Troup, S. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I. Kim, A. Soboleva, M. Tomashevsky, and R. Edgar, NCBI GEO: mining tens of millions of expression profiles database and tools update, *Nucleic Acids Research*, **35** (2006), D760–D765.
[56] L. Stein, Creating a Bioinformatics Nation, *Nature*, **417(6885)** (2002), 119–120.
[57] http://www.ncbi.nlm.nih.gov/.
[58] http://www.hapmap.org/.
[59] http://www.biobase-international.com/pages/.
[60] https://www.pharmgkb.org/.
[61] W. Kent, C. Sugnet, T. Furey, K. Roskin, T. Pringle, A. Zahler, and D. Haussler, The Human Genome Browser at UCSC, *Genome Research*, **12(6)** (2002), 996–1006.
[62] A. Su, et al., Large-scale analysis of the human and mouse transcriptomes, National Academy of Sciences of the United States of America (2002).
[63] V. Matys, et al., TRANSFAC: transcriptional regulation, from patterns to profiles, *Nucleic Acids Research*, **31** (2003), 374–378.
[64] M. Friedman, A. Levy, and T. Millstein, Navigational plans for data integration, Proc. 16th National Conference on Artificial Intelligence (AAAI), 1999.
[65] R.L. Elsberry, T.D.B. Lambert, and M. Boothe, Accuracy of atlantic and eastern north pacific tropical cyclone intensity forecast guidance. Submitted to *Weather and Forecasting* (2006).

[66] F.D. Marks, and L.K. Shay, Landfalling tropical cyclones: Forecast problems and associated research opportunities, *Bull. Amer. Met. Soc.,* **79** (1998), 305–323.
[67] http://www.wrf-model.org/index.php.
[68] http://www.openmp.org/drupal/.
[69] H.E. Willoughby. Improvements in observations, models and forecasts. *HURRICANE! Coping with Disaster*, R.H. Simpson, Ed., AGU (2002), 205-216.
[70] P. Singh, N. Zhao, S.-C. Chen, and K. Zhang, Tree Animation for a 3D Interactive Visualization System for Hurricane Impacts, *Proc. IEEE Intl. Conf. on Multimedia* (2005), 598-601.
[71] S.-C. Chen, S. Hamid, S. Gulati, N. Zhao, M. Chen, C. Zhang, and P. Gupta, A Reliable Web-based System for Hurricane Analysis and Simulation, *Proc. IEEE International Conference on Systems, Man and Cybernetics* (2004), 5215–5220.
[72] S.-C. Chen, M.-L. Shyu, C. Zhang, W.Z. Tang, and K. Zhang, Damage Pattern Mining in Hurricane Image Databases, *Proc. IEEE Intl. Conf. on Info. Reuse and Int.* (2003), 227–234.
[73] O. Ezenwoye and S.M. Sadjadi, Composing aggregate web services in BPEL, *Proc. 44th ACM Southeast Conference* (ACMSE), March 2006, Melbourne, FL.
[74] O. Ezenwoye and S.M. Sadjadi, TRAP/BPEL: A framework for dynamic adaptation of composite services, *Proc. International Conference on Web Information Systems and Technologies* (WEBIST), March 2007, Barcelona, Spain.
[75] O. Ezenwoye and S.M. Sadjadi. RobustBPEL2: Transparent autonomization in business processes through dynamic proxies, *Proc. 8th International Symposium on Autonomous Decentralized Systems* (ISADS), March 2007, Sedona, AZ.
[76] O. Ezenwoye and S.M. Sadjadi, Enabling robustness in existing BPEL processes., *Proc. 8th International Conference on Enterprise Information Systems*, May 2006, Paphos, Cyprus.
[77] S.M. Sadjadi, J. Martinez, T. Soldo, L. Atencio, R.M. Badia, and J. Ejarque, Improving separation of concerns in the development of scientific applications. *Technical Report FIU-SCIS-2007-02-01*, School of Computing and Information Sciences, Florida International University, Miami, FL, February 2007.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Computer Science Grids

Franck CAPPELLO [a] Henri BAL [b]
[a] *INRIA, France*
[b] *Vrije Universiteit, The Netherlands*

Abstract. The Computer Science discipline, especially in large scale distributed systems like Grids and P2P systems and in high performance computing areas, tends to address issues related to increasingly complex systems, gathering thousands to millions of non trivial components. Theoretical analysis, simulation and even emulation are reaching their limits. Like in other scientific disciplines such as physics, chemistry and life science, there is a need to develop, run and maintain generations of scientific instruments for the observation of complex distributed systems running at real scale and under reproducible experimental conditions. Grid'5000 and DAS3 are two large scale systems designed as scientific instruments for researchers in the domains of Grid, P2P and networking. More than testbeds, Grid'5000 and DAS3 have been designed as "Computer Science Grids", where researchers share experimental resources spanning over large geographical distances, are able to reserve resources, configure them, run their experiments, realize precise measurements and replay the same experiments with the same experimental conditions. Computer scientists use these two platforms to address issues in the different software layers between the hardware and the users: networking protocols, OS, middleware, parallel and distributed application runtimes, and applications. In this paper, we will present two Computer Science Grids: Grid'5000 and DAS3. We will describe the motivations, design and current status of these two systems. We will also present some of their key results, not only in terms of scientific results in computer science, but also the impact of these two systems as research tools. The success of the Grid'5000 and DAS platforms is the basis of an international initiative, having the objective to deploy a European level "Computer Science Grid".

Keywords. Computer Science Grid, large scale experiments, reconfiguration, controlable experimental conditions

Introduction

Grid and P2P systems are very popular as production platforms (EGEE, TeraGrid, SETI@home, Edonkey, Skype) and inspire a wide spectrum of research. These distributed systems are still difficult to design, operate and optimize due to their software complexity, heterogeneity, the volatility of their components and their large scale. As a consequence, many institutes and international programs develop significant funding efforts to foster Grid and P2P research initiatives.

As a matter of fact, the research in Grid and P2P systems span over all the layers of the software stack between the user and the hardware. The applications, programming environment, runtime systems, middleware, operating systems and networking layers are subject to extensive studies seeking to improve their performance, security, fairness, robustness and quality of service.

Like other scientific domains, research in Grid and P2P computing is based on a variety of methodologies and tools. When Grid'5000 [1] and DAS were designed, most of the research conducted in Grids and P2P systems was performed using simulators [2] [3] [4], emulators [5] or production platforms. However, all these tools present limitations making the study of new algorithms and optimizations difficult. Simulators focus on a specific behavior or mechanism of the distributed system and abstract the rest of the system. A main restriction of simulators is the difficulty of their validation. Indeed very few studies have been conducted to validate the existing simulators. When it becomes difficult to capture and extract the factors influencing the distributed systems, emulators can help by executing the actual software of the distributed system, in its whole complexity, on a fully controlled platform. As a consequence, there is still a gap between emulators and the reality: they cannot capture all the dynamic, variety and complexity of real life conditions.

Production platforms may be considered as good candidate for experimentation because they expose the software to experiment to realistic conditions. However, there are several reasons why computer scientists require their own infrastructure and cannot use existing production machines. Foremost, many computer science projects require experiments with the operating system and communication protocols, for example to investigate various resource management policies or security mechanisms within the operating system. Some projects even require the installation of experimental hardware, such as advanced network interface cards or the GRAPE N-body processors. These types of experiments are hard to do on production machines. Secondly reproducing the experimental conditions several times is almost impossible on production platforms. Thirdly, there is a clear difference between how computer scientists and application scientists use the resources. Application scientists just want to run large experiments that take much compute time, often many 100,000s node hours per year. Most production machines are thus optimized for high job throughput and have a high utilization degree. Production applications typically run on a single site, and many applications even consist of a large number of sequential jobs that run on a single node. Several projects allow sharing of multiple production machines, but each application typically runs on a single site, determined by a global scheduler. Computer scientists, on the other hand, want to do distributed experiments that run on many sites at the same time, for example to do research on grid middleware, P2P systems, or distributed algorithms. Also, the experiments are far more interactive and large-scale: they run on many nodes, but for a relatively shorter time. This requires optimizing the system for fast job-startups rather than for high throughput. The utilization degree of the system should be low during daytime, to allow successful co-allocation of multiple resources at the same time.

Thus, the complexity of Grid and P2P systems raise the need for real-scale experimental platforms where computer scientists can run experiments, observe the distributed systems at large scale, stress the systems using experimental conditions injectors and make precise measurements. In theory, such platforms could be built by combining existing clusters from different universities. Such ad hoc grids are used in many European projects. This approach has major disadvantages. It is inefficient, because each project has to invest much manpower into setting up such an infrastructure. Moreover, the existing clusters are (by definition) also used for other purposes, so each time they are needed for grid experiments much effort is needed to prepare them. Often, cooperation from a local system administrator is needed (e.g. for changing configuration settings), so

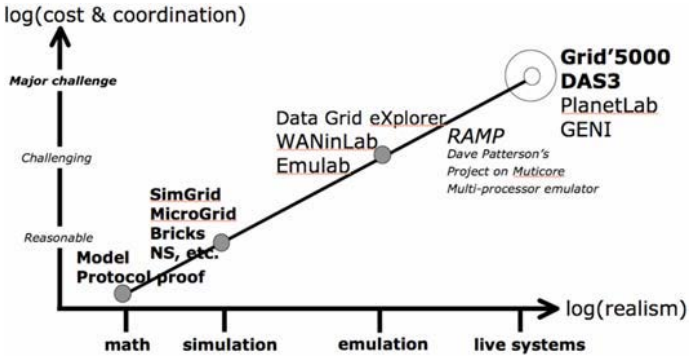


Figure 1. Methodologies used in distributed system studies.

the preparation time increases with the number of resources. In practice, this situation is highly ineffective.

Grid'5000 and DAS were designed by a community of computer science researchers. More than testbeds, Grid'5000 and DAS have been designed as Computer Science Grids, where researchers share experimental resources spanning over large geographical distances, are able to reserve resources, configure them, run their experiments, realize precise measurements and replay the same experiments with the same experimental conditions. We may consider them as large scale research instruments like particle accelerators or telescopes for physicists and astrophysicists; the main difference being the user community of Grid'5000 and DAS which is the researchers in computer science. Grid'5000 and DAS avoid having remote humans in the loop and make sure the systems always are available for distributed experiments. This need to develop and share common instruments is shared by computer architects. The RAMP project developed by several universities in the USA has many methodological motivations similar to the ones which guided the design of Grid'5000 and DAS: speed, malleability, observability, reproducibility, and credibility.

Figure 1 presents the spectrum of methodologies used by researchers to study issues in distributed systems and especially in Grid and P2P systems. Note that PlanetLab [6]sits in between a production platform and a real-scale experimental platform since its edge resources (PCs) are not fully controlled by the users and the networking conditions are not reproducible.

The cost (vertical axis) is an expression of the technical difficulty, actual hardware, maintenance and running costs and the complexity to manage and operate the research tool. One of the lessons learned during the first years of the Grid'5000 project is the complexity of coordinating a team of engineers distributed geographically to develop, run and maintain the platform hardware and software components.

In this paper we thus present two computer science grids: Grid'5000 and DAS. Grid'5000 is substantially larger and has more advanced configuration support. DAS has a longer history, including DAS1 (1997), DAS2 (2002), and DAS3 (2006), and is designed as a simple but very reliable testbed for computer scientists. In Section 2, we present the design principles of Grid'5000 and DAS3, which are the results of the experiences of Grid researchers. The implementation and status of Grid'5000 and DAS is presented in Section 3. Section 4 presents examples of key results in Grid, P2P and Networking obtained with the two platforms. In Section 5 we present the impact of Grid'5000 and

DAS as research instruments, giving statistics such as the number of users and the number of papers published using them. As conclusion, in section 6 we present the initiative toward a European scale Computer Science Grid.

1. Design

The designs of Grid'5000 and DAS3 were established separately. Both platforms are the results of 1) the past experiences on testbeds for Grid research such as eToile in France and DAS1 and DAS2 in the Netherlands, and 2) the description by the computer scientists of their needs in experimentation. These two elements led to propose large scale experimental platforms, with deep reconfiguration capabilities and a strong control and monitoring infrastructure.

1.1. Grid'5000

During the preparation of the project in 2003, we conducted an analysis on the need of a computer science Grid and the diversity of potential experiments. The researchers of the Grid computing community in France, involved in many French ACI Grid projects and European Grid projects, proposed a set of about 100 experiments. A first conclusion of the analysis was the need for a large scale (several thousands of CPUs), distributed (10 sites) computer science Grid. A second conclusion was that the experiment diversity nearly covers all layers of the software stack used in Grid computing, from the user interface to the networking protocols. A third conclusion was that most of the researchers need a specific experiment setting, different from the other researchers. Researchers involved in networking protocols, OS and Grid middleware research often require a specific OS for their experiments. Some research on virtual machines, process checkpointing and migration need the installation of specific OS versions or OS patches that may not be compatible. Researchers needs are quite diverse in Grid Middleware: some require Globus, while others need Unicore, Desktop Grid or P2P middleware. Some other researchers need to test applications and mechanisms in a multi-site, multi-cluster environment, without any Grid middleware.

As a consequence, we concluded that Grid'5000 should provide a deep reconfiguration mechanism allowing researchers to deploy, install, boot and run their specific software images, possibly including all the layers of the software stack. This reconfiguration capability led to the experiment workflow followed by Grid'5000 users: 1) reserve a partition of Grid'5000, deploy a software image on the reserved nodes, reboot all the machines of the partition using the software image, run the experiment, collect results and relieve the machines.

Because researchers are able to boot and run their specific software stack on Grid'5000 sites and machines, we decided 1) to isolate Grid'5000 from the rest of the Internet and 2) to let packets fly inside Grid'5000 without limitation. The first choice ensures that Grid'5000 will resist to hacker attacks and will not be used for Internet attacks. The second choice guarantees that communication performance does not suffer from the overhead of an imposed security system. Thus, Grid'5000 is built as a large scale confined cluster of clusters. Strong authentication and authorization checks are done when users log in Grid'5000.

Grid'5000 is composed of heterogeneous resources. However, we decided to keep at least 2/3 of the machine homogeneous in Grid'5000 for two main reasons: 1) speedup evaluation is difficult to evaluate with heterogeneous hardware, 2) hardware diversity increases the complexity of the deployment, reboot and control subsystems and the every day management and maintenance cost.

The capability to reproduce experimental conditions is fundamental in experimental tools, especially when performance comparisons are conducted. To fulfill this strong requirement, we decided to use dedicated network links between sites, to allow users reserving the same set of resources across successive experiments, to allow users running their experiments in dedicated nodes (obtained by reservation) and to let users install and run their proper experimental condition injectors and measurements software. Thus every user has full control of the reserved experimental resources.

1.2. DAS3

DAS was designed to be a distributed infrastructure that is shared between many computer science groups of the ASCI research school. ASCI (Advanced School for Computing and Imaging) is a formal collaboration between several Dutch universities. Three systems have been built so far: DAS1 (by Parsytec, 1997), DAS2 (IBM, 2002), and DAS3 (ClusterVision, 2006). Each system consisted of 4-5 clusters. The major design principle behind the DAS systems is to make them robust by keeping the design as simple as possible. An important design choice therefore is to make the systems homogeneous: all clusters use the same CPU type, network (Myrinet), and operating system. This allows clean, laboratory-like experiments and eases system administration and exchange of software between different universities.

The most novel part of the new DAS3 system is its optical wide-area interconnect (see Figure 2), which is provided by SURFnet, as part of its Gigaport project. An extra band of the optical SURFnet-6 network is allocated to DAS3, with up to 8 lambda's of 10 Gb/s each, giving DAS3 its own dedicated optical network, separate from the university backbone. DAS does not support the high degree of reconfiguration that Grid'5000 allows, but DAS3 was designed to allow reconfiguration of the optical wide-area network. Using Wavelength Selective Switches (WSS), the topology of the optical interconnect can be changed dynamically, giving DAS3 a unique reconfigurable optical network, called StarPlane (see www.starplane.org).

2. Organization and Status

The construction of Grid'5000 started in 2004 while that of DAS3 began in mid 2006. As a consequence, the status of the two systems are different: Grid'5000 software tools, while still in evolution, are quite mature and robust. The users and administrators have acquired 2 and 1/2 years of experience in using Grid'5000, preparing and running experiments.

2.1. Grid'5000

Based on the design decisions presented in the previous section, we decided to build a platform of 5000 CPU-cores distributed over 9 sites in France. Figure 3 presents an

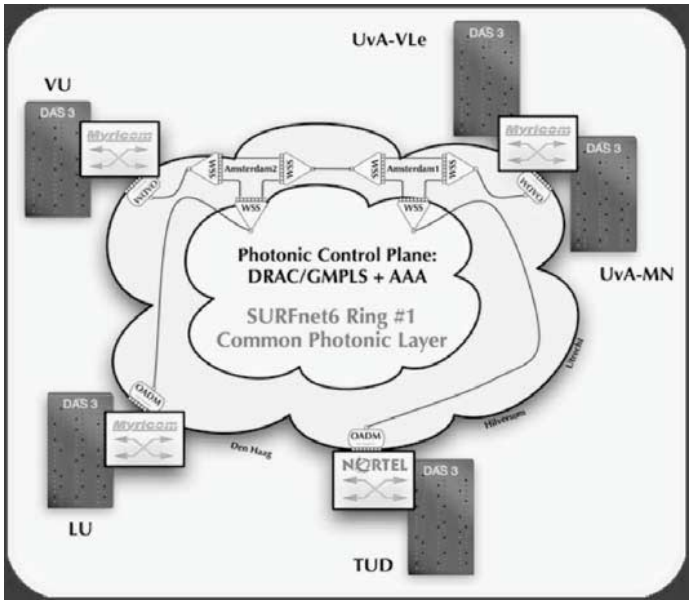


Figure 2. DAS3 with its optical wide-area network.

overview of Grid'5000. Every site hosts a cluster and all sites are connected between each others by high speed network links (RENATER 4: 10 Gbps links).

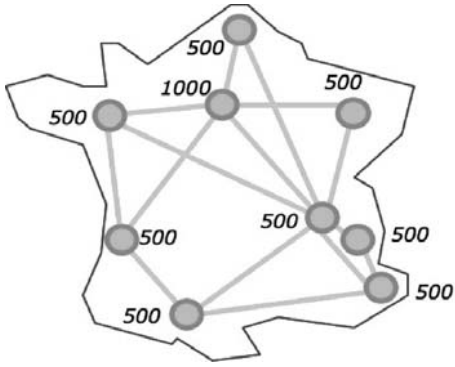


Figure 3. Overview of Grid'5000.

Numbers in Figure 3 give the target number of CPUs for every cluster. 2/3 of the nodes are dual CPU 1U racks featuring 2 AMD Opteron running at 2 Ghz, 2 GB of memory and two 1Gbps Ethernet Adapters. Clusters are also equipped with high speed networks (Myrinet, Infiniband, etc.). Disk space varies across the 9 sites. Most of the CPUs racks provide at least 80 Gbytes of local non archived storage. In addition, all Grid'5000 sites provide more than 1 Tbybtes of storage (replicated or archived) for the user experimental data.

Every user has a single account on Grid'5000. Every Grid'5000 site manages its own user accounts and runs an LDAP server containing the same tree: under a common

root, a branch is defined for each site. On a given site, the local administrator has read-write access to the branch and can manage its user accounts. The other branches are periodically synchronized from remote servers and are read-only. Once the account is created, the user can access any of the Grid'5000 sites or services (monitoring tools, wiki, deployment, etc.). User data are kept local to every site and distribution to remote sites is done by the user through classical file transfer tools (rsync, scp, sftp, etc.). Data transfers from and to the outside of Grid'5000 are restricted to secure tools and done on gateway servers.

At cluster level, users submit their resource reservations and experiment jobs using the OAR [7] reservation engine and batch scheduler. OAR provides most of the important features implemented by other batch schedulers such as priority scheduling by queues, advance reservations, backfilling and resource match making. OAR relies on a specialized parallel launching tool named *Taktuk* [8] to manage all large-scale operations like parallel tasks launching, nodes probing or monitoring. At the Grid level (Cluster of Clusters) a simple broker collocates the resources of several Grid'5000 sites by submitting reservations to the local OAR schedulers. Currently, if one reservation is refused, all previously accepted reservations are canceled. This simple meta-reservation approach is acceptable when the platform workload is moderate. Clearly it should be replaced by a more sophisticated approach when normal workload leads to many meta reservation cancellations.

To reconfigure the software stack on every reserved node, the users run the Kadeploy2 [9] deploying the user defined software environment on a disk partition of selected nodes. The software environment contains all software layers from the OS to application, in addition to experimental condition injectors and measurements tools. Deployment begins by rebooting all nodes on a minimal system through a network booting sequence. This system prepares the target disk for deployment (disk partitioning, partition formatting and mounting). Then the environment is broadcast to the selected nodes using a pipelined transfer with on the fly image decompression. At this point, some adjustments must be done on the broadcasted environment in order to be compliant with node and site policies (mounting tables, keys for authentication, information for specific services that cannot support auto-configuration). The last deployment step consists in rebooting the nodes on the deployed system from a network loaded bootloader.

2.2. DAS

DAS1 and DAS2 were primarily designed for experiments with algorithms, programming environments and systems software, for which such a clean design turned out to be a major advantage. For DAS3, the research focus shifted more towards designing grid middleware and e-Science software (scheduling, workflow, visualization, etc.), with an increasing need for a somewhat more heterogeneous testbed.

Therefore, the DAS3 clusters have different CPU speeds, a mix of single- and multi-core nodes, and some variation in local network (one site uses only Ethernet, the others also have Myrinet). The types of the CPUs and the operating system (Linux) are uniform, to significantly ease system administration. DAS3 has 272 nodes (2.2-2.6 GHz dual-SMP AMD Opterons), each with 4 GB memory (over 1 TB in total), and 250-500 GB local disks (84 TB in total). One of the clusters will be extended shortly with 47 TB of extra disk space, to allow experiments with large-scale multimedia data in the MultimediaN project.

Each cluster has 2-10 TB storage on its head node (29 TB in total). This mass storage at every DAS3 site is NFS-mounted on the local compute nodes and subdivided into two categories: user data that is backed up, and a large amount of scratch space, containing for example copies of large external data sets and temporary results. Scratch space is not backed up: the sheer amount of mass storage forced us to make this choice, which works out well in practice. Users are themselves responsible to transfer or replicate their data over different sites, if needed by their application. In principle, the DAS3 wide area interconnect has a large enough bandwidth to facilitate running a DAS3-wide distributed file system, but this is currently not implemented. Note that spurious file system traffic would interfere with true wide area application traffic, which makes performance analysis more difficult.

DAS1 used a dedicated 6 Mb/s ATM wide-area network between the clusters; DAS2 used the university backbone network. The optical SURFnet-6 network gives DAS3 a much better balance between the computing and networking power. The optical network is connected to the clusters using the latest 10 Gbit/s version of Myrinet, Myri-10G. Like previous Myrinet generations (which were also used in DAS1 and DAS2), Myri-10G offers a state-of-the-art low latency, high throughput interconnect, but what makes it especially attractive for DAS3 is its native compatibility with 10 Gbit/s Ethernet. The Myri-10G switches at the DAS3 sites contain special blades that take care of the rather straightforward 10G Ethernet/Myrinet protocol translation (low-level flow control and packet routing). The net effect is that a DAS3 compute node with a Myri-10G NIC can directly communicate with both local nodes as well as off-site resources at full 10G speed, without any intermediate layer-3 routers. This can be very advantageous for several high performance distributed application scenarios on a Grid.

DAS uses a simple system administration model that is coordinated from a central site (VU Amsterdam). It uses a single, replicated, user account file. Likewise, the security model is fairly simple. Since the DAS clusters are used for truely distributed computing, often requiring direct coupling with remote computing and data resources, applying highly restrictive firewalls at each site would hamper research and daily operations too much. However, a reasonable security level is obtained by restricting the number of system services with open ports to a minimum, and improved by allowing direct login via SSH only to the head nodes of the clusters from a limited list of sites or hosts. Jobs on the compute nodes can only be started from a site's head node. Jobs arriving via the Globus grid infrastructure are only accepted for users that obtained their certificate from certificate authorities implementing well-accepted security procedures.

The cluster and Grid systems software used in DAS likewise evolved. For DAS1, we used no true Grid middleware, and local compute nodes were allocated using a simple, home-brew scheduler. Cross-site jobs were typically started using shell scripts. For DAS2 and DAS3, we use the Globus Toolkit, in order to adhere to international standards, which is especially important for large projects like VL-e (see below). DAS2 and DAS3 employ the standard SGE (Sun Grid Engine) cluster resource management system, which has proven to be very reliable over the years. As currently set up, every cluster has its own, independent, SGE scheduler which can be used both through local interfaces or via Globus. Policies related to cross-site scheduling are implemented using additional services. For example, the KOALA [10] software developed at TU Delft supports co-allocation of multiple resources, possibly taking cluster load and proximity of data sets into account. Also, many of our projects now use higher-level APIs like the

Grid Application Toolkit (GAT) or the new Simple API for Grid Applications (SAGA) to abstract from the specifics of the underlying Grid middleware.

DAS3 is running standard SL4 (Scientific Linux version 4) but the kernel is replaced with one based on the recent version 2.6.18 to obtain peak TCP/IP performance (over 9 Gbit/s) over Myri-10G. At 10 Gbit/s implementing the networking stack and interrupts efficiently (e.g., using PCI's Message Signaled Interrupts) becomes an important issue.

At present, all clusters for DAS3 are operational. Setting up the optical wide-area network required major efforts and investments from SURFnet. This interconnect is expected to be operational soon.

3. Examples of key results

Grid'5000 and DAS have already produced significant results in computer science for the community of Grid and P2P researchers. This part presents the domains covered by the experiments in the two platforms.

3.1. Grid'5000

The main objective of Grid'5000 and its associated software set is to ease the deployment, execution and result collection of large scale Grid experiments. Currently about 300 experiments are planned or realized. The topic of these experiments cover all the layers of the software stack between the user and the Grid resources.

About 50 experiments are planned at the networking layer. This includes research on high speed protocols, monitoring, distributed measurements, high performance protocols for MPI on the Grid, high bandwidth data transfer analysis and modeling, traffic isolation, stress of 10G WAN links, realistic Internet traffic replay, Grid collective communications, transfer time prediction, etc.

More than 100 experiments are proposed for the middleware layer. This set of experiments includes tests on Globus, OGSA-DAI, fault tolerant MPI, distributed storage systems, automatic Grid infrastructure deployment, rapid and dynamic virtual cluster creation, Desktop Grid environments, data management and scheduling in Desktop Grids, P2P DHT, meta and hierarchical Grid schedulers, fully distributed batch schedulers, automatic Grid execution checkpointing, JXTA performance and scalability, resource discovery systems, etc.

About 40 experiments are planned for the programming layer of the software stack. For this layer, the research concerns the design, implementation, tests and evaluation of Grid programming environments, such as Workflow description and runtime tools (YML, OpenWP, etc.), Grid versions of MPI implementations (MPICH and OpenMPI), Grid RPC environment such as DIET and OmniRPC, combinatorial optimization environment such as PARADISEO-G, Object oriented parallel and distributed computing in Java with ProActive, Component model environments, etc.

The application layer receives a strong interest with more than 80 experiments (done or planned). The main purpose of this research is to evaluate the performance of applications ported on the Grid and test alternatives or design new algorithms and new methods for these applications. The application domains cover life sciences (mammograms comparison, protein sequencing from tandem mass spectrometry, gene prediction, virtual

screening funnel, conformation sampling and docking, etc), physics (seismic imaging, parallel solvers for two phase flows, hydrogeology, simulation of self-propelled solids in a viscous incompressible fluid, Particle Image Velocimetry, seismic tomography, geophysical inverse problem, climate modeling, 3D discrete ordinates neutron transport: SWEEP3D, fluid mechanics, external aerodynamics, radiative transfer coupled to hydrodynamics, etc.), applied mathematics (sparse matrix computation, combinatorial optimization solvers, parallel and distributed model checkers, PDE problem solving with asynchronous iterations, etc.), chemistry (molecular simulation, estimation of thickness and optical constants of thin films, etc.), industrial processes, financial computing, etc.

In addition there is a set of 20 experiments that do not fit in one of the previously mentioned layers. They concern operating systems (XtremOS), virtualization techniques and software tools to be used as Grid'5000 mechanisms: heterogeneity emulators, experimental condition injectors, monitoring tools, fast software deployment and reconfiguration tools, etc.

To highlight the interests and benefits of using Grid'5000, we present three experiment examples in three different layers of the software stack.

The first example concerns the middleware layer. This is collaborative work with Sun Microsystems about the scalability of the JXTA P2P framework. The experiments evaluated two main protocols of JXTA [11]: 1) the peerview protocol used to organize super peers, known as rendezvous peers, in a JXTA overlay and 2) the discovery protocol, relying on the peerview protocol, used to find resources inside a JXTA network. All sites of Grid'5000 were used and a mix of hundreds of rendezvous peers and normal peers (called edge peers) were deployed on at most 580 nodes. The goal of the evaluation is to answer common and unanswered questions about JXTA scalability: how many rendezvous peers are supported by JXTA in a given group and what is the expected time to discover resources in such groups? Results show that with the default configuration of the peerview protocol, the algorithm does not scale, even with as few as 45 rendezvous peers. Larger configurations in terms of number of rendezvous peers are still possible after careful tuning. For the discovery protocol, the experiment demonstrated that discovery time is rather small, provided that local peerviews are consistent across all rendezvous peers of a given group. These results clearly demonstrate that even for industrial software with production quality, there is a strong need to test and evaluate the properties of the distributed system in real scale platform such as Grid'5000.

The second example concerned the programming environment layer and combinatorial optimization algorithms. Optimally solving large instances of combinatorial optimization problems using a parallel Branch and Bound (B&B) algorithm requires a huge number of computational resources. In [12], the authors proposed a gridification of the parallel B&B algorithm, based on new ways to efficiently deal with some crucial issues, mainly dynamic adaptive load balancing, fault tolerance, global information sharing and termination detection of the algorithm. A new efficient coding of the work units (search sub-trees) distributed during the exploration of the search tree is proposed to optimize the involved communications. The algorithm has been implemented following a large scale idle time stealing paradigm (Farmer-Worker) and experimented on the Flow-Shop NP-hard scheduling problem instance ($Ta056$) (scheduling of 50 jobs on 20 machines). The algorithm allowed to improve the best known solution by providing the optimal solution with proof of optimality. The problem was solved within 25 days using about 1900 processors belonging to 6 clusters of the Grid5000 and to 3 clusters from Université de

Lille1 . During the resolution, the worker processors were exploited with an average of 97% while the farmer processor was exploited only 1.7% of the time. These two rates are good indicators of the efficiency of the proposed approach and its scalability. This result can be considered as a success story since the problem instance has never been solved exactly before.

The third example concerns the application level. Since the last 30 years, many research works in geophysics (seismology) focused on seismic tomography to reveal the structure of Earth interior. To solve the resolution limitation of tomographic models, an irregular model is used, which adapts locally to the density of seismic information. This method computes the seismic tomography from the huge amount of data, based on the seismicity of the world from the years 1964 to 1995 (approximately 82000 seisms and 12 millions of rays, about 1.2 millions significant rays after pre-processing). To speed-up the computation, several specific parallel MPI programs have been developed. The experiment [13] concerned the first step of a method, which consists in ray-tracing the seismic rays (the waves' paths) from the recorded seismic events. This step is highly parallel since every ray can be traced independently. However the method eventually requires an all-to-all communication phase, which is a real bottleneck on many hardware platforms. In July 2006, Grid'5000 was used for a tomography using the full dataset. Several configurations were tested to assess the application scalability, with 32, 64, 128, 192 and up to 458 processors, on 1, 2, 3 or 5 sites. Despite a considerable volume of data exchanged in the all-to-all phase (7 GB, 15 GB and 20 GB for 32, 128 and 458 processors resp.) the speedup stays nearly linear. Moreover the application performance does not significantly decrease when using 3 sites instead of 2. The global method takes 227s on 458 processors, 3164s on 32 processors (a single cluster) and more than 36 hours on a single PC. This experiment demonstrates: 1) this class of applications scales extremely well on Grid'5000, 2) MPI applications can run efficiently on a Grid, 3) a platform like Grid'5000 is a very useful tool to evaluate the scalability and performance of parallel applications on the Grid.

3.2. DAS

Many important computer science results have been obtained on the DAS systems over the past 10 years, as summarized briefly below. A good indication of the research performed with DAS are the Ph.D. theses that used the system for their experiments. An overview and classification of these theses is given in the table below.

Much research has been done on programming environments for cluster computers, including various distributed shared memory systems (like the Orca language), efficient communication in Java, and communication protocols for Myrinet. Also, several compiler projects have used (or are still using) DAS.

After this work on cluster computing, the focus shifted more towards grid computing. Successful experiments have been done with running non-trivially parallel programs on a wide-area system, for applications like search algorithms, N-body simulations, SAT-solvers, image processing, weather modelling, video analysis, astronomy [14], and many others. This work has shown that distributed supercomputing on grids is more generally applicable than just high-throughput (trivially parallel) applications. Many effective optimizations for distributed supercomputing have been found, including load balancing mechanisms, latency hiding techniques, and algorithmic improvements.

Parallel programming systems:
Communication Architectures for Parallel-Programming Systems
Collective Computation in Object-based Parallel Programming Languages
Java for High Performance Computing
MultiGame: An Environment for Distributed Game-Tree Search
Compiler and Runtime Optimizations for Fine-Grained DSM Systems
Method Invocation Based Communication Models for Parallel Programming in Java
Compile-Time Scheduling for Distributed-Memory Systems
Compilers for HPC:
Iterative Compilation in Program Optimization
Rule-based Compilation of Data-Parallel Programs
Statistical compiler tuning
Code Generation for Large Scale Applications
Grid computing:
Efficient Java-Centric Grid-Computing
Performance Analysis of Processor Co-Allocation in Multicluster Systems
Time Warp - from Cluster to Grid
Handling complexity and change in grid computing
Distributed systems:
An Object-Based Software Distribution Network
Locating Objects in a Wide-Area System
An Approach to a Scalable Wide-Area Web Service
Epidemic-Based Self-Organization in Peer-to-Peer Systems
Separation and Adaptation of Concerns in a Shared Data Space
Scientific Information in Collaborative Experimental Environments
Applications:
Branching Growth in Stony Corals
Hybrid Systems for N-Body Simulations
Mesoscopic Computational Haemodynamics
Simulating self-gravitating systems on parallel computers
Grid-based HLA Simulation Support
Plan Merging in Multi-Agent Systems
Agent-Based Matchmaking and Clustering
An agent based architecture for constructing Interactive Simulation Systems
Improving Visual Matching; Similarity Noise Distribution and Optimal Metrics
User Transparent Parallel Image Processing
Data and Task Parallelism in Parallel Image Processing Applications
Complex Streamed Media Processor Architecture

Table 1. Titles of Ph.D. theses done using DAS2

New programming systems have been developed that are especially designed for grid computing [15], such as the Ibis Java-centric system, the Satin divide-and-conquer system, the MagPIe and Dynamite MPI libraries and the Spar language. Several of these systems have been used in large-scale experiments, often in combination with Grid'5000. A new scheduler (KOALA) was developed that is able to co-allocate multiple resources (sites) at the same time, and to optimize co-allocation by taking the locations of large input files (and their replicas) into account. This research extensively uses tracefiles from

DAS itself to study the performance of various scheduling algorithms [10].

An important related topic is distributed systems, which addresses issues like replication of web pages, self-organization, and distribution networks. DAS was used extensively to study the interaction of new high speed Internet transport protocols in varying mixes with standard TCP over different international optical connections [16].

DAS is also used to investigate advanced applications, for example from computational science, astronomy, imaging, and agent technology. Also, many novel interactive application experiments have been performed. For example, in the European CrossGrid project, a Grid-enabled, distributed vascular bypass operation has been implemented and demonstrated. The demonstration consists of a CT-scanner connected with a blood-flow simulator (running on DAS2) and a virtual operation. One other result that obtained much interest from the media was the fact that John Romein (VU) solved the 3000 year old game of Awari[17]. He used a parallel retrograde analysis program running on the 144-CPU DAS2 cluster at the VU, showing that the game-theoretical value of the game is a draw.

The new DAS3 system will be used for many other projects in these areas, and also for two very large (30-40 M. Euro) research programs: VL-e (Virtual laboratory for e-Science) and MultimediaN. The computer science research in VL-e focuses on interactive problem solving environments, workflow, information management, adaptive information disclosure, visualization, and grid computing.

We discuss three research projects for DAS3 in more detail: Ibis, KOALA, and JADE-MM.

Ibis is a Java-centric grid programming system that exploits Java's 'write once, run anywhere' portability to run applications on heterogeneous grids. Ibis has been used to develop parallel Grid applications that run on large numbers of resources distributed across Europe, including DAS3 and Grid'5000. Ibis has been used to implement several high-level programming models. In particular, our divide-and-conquer system (Satin) is highly suitable for writing parallel Grid applications. Using DAS, we have shown that this model can be made fault-tolerant and can even automatically adapt to changing conditions in the grid, like overloaded CPUs or networks [18]. Ibis and Satin are used by several institutes, for applications like analyzing brain images, protein identification, and grammar learning.

KOALA is a grid scheduler which has as its aim to provide a transparent scheduling service across multiple clusters in a grid. KOALA's two most important features are its support for co-allocation (i.e., for allocating processors in more than one cluster to single applications), and for load sharing across clusters. In addition, it supports file movement and Ibis applications that require co-allocation. With KOALA, we have shown that it possible to design, implement, test, and deploy a real grid scheduler that employs efficient scheduling policies and that gives a reliable scheduling service to daily users. Furthermore, we have designed and implemented tools such as Grenchmark for grid performance evaluation.

Part of the MultimediaN project focuses on emerging problems in the field of *multimedia content analysis*, ranging from real-time comparison of objects and individuals in video streams obtained from surveillance cameras, to interactive access to archives of current and historic television broadcasts. As digital video may produce high data rates, and multimedia archives steadily run into Petabytes of storage space, compute power is a persistent bottleneck for multimedia content analysis [19]. To provide researchers in the

multimedia domain with efficient and transparent access to Multimedia Grid services, much research is required — especially in the definition of easy-to-use programming models for non-experts in Grid computing, in the automatic optimization of resource utilization, and in the performance modeling of heterogeneous systems.

To this end, our JADE-MM project aims, among other things, to develop stochastic control schemes that make time-constrained multimedia applications tolerant to the dynamics of large-scale Grid environments. Recent results obtained on DAS-2 and DAS-3 [19] have shown convincingly that Grid-based distributed computing can bring efficient solutions for state-of-the-art multimedia problems. One example is a real-time visual object recognition task performed by a robot. Our solution to this computationally demanding problem, requiring up to 20 seconds *per video frame* on a fast sequential machine, is one of the few known instances of a real-time task involving extensive communication patterns among geographically distributed resources. For this application we have obtained a 'most visionary research award' at AAAI 2007 - one of the most high-profile conferences in the field of artificial intelligence.

Another example constitutes our participation in the international NIST TRECVID evaluation, i.e. the yearly benchmark for the comparison of approaches to finding semantic concepts (e.g., human faces, cars, interviews) in archives of news broadcasts from ABC and CNN. Our algorithmic approaches, in combination with the use of DAS-2 and DAS-3, have brought decisive advantages, resulting in multiple 'best performances' in a field of strong international competitors (a.o. including IBM Research and Carnegie Mellon University). For this off-line application, which would require over 10 years of processing on the fastest sequential machine at our disposal, we have also obtained a 'best technical demo award' at ACM Multimedia 2005. Clearly, with data volumes commonly in the peta-scale range, and the presence of extensive and irregular communication patterns, DAS-3 is essential for this line of research.

4. Impact as research instruments

Designing, constructing and running a Computer Science Grid raises many technical issues and has a significant cost. Grid'5000 and DAS3 should be evaluated as research tools, the quality and quantity of the scientific results they have produced and their impact on the research community.

4.1. Grid'5000

One of the most significant signs of success of Grid'5000 is its number of users. We currently have about 250 active users who present their experiment context and report on the Grid'5000 web site. We frequently receive requests from foreign colleagues to get an account and use Grid'5000, despite the fact that Grid'5000 access is rather restricted, because all funding is supported by France. Foreign colleagues can get an access to Grid'5000 through collaborations with a French research team. This is the case for the participants of several European projects of the European Frame Work program 6 (Grid4all, QosCos, XtreemOS, etc.). In total, the users are from 60 computer science laboratories worldwide.

Beyond the attractiveness, the main result is the number of publications: in about 2 years of exploitation, Grid'5000 has been used for 4 HDR (a diploma in France that

could be obtained 4 or 5 years after the Ph. D.), a dozen of Ph. D., tens of Master theses and hundreds of publications. We continuously observe an increase of the number of master students, Ph. D. candidates and researchers using it.

A nice measure of the Grid'5000 usefulness is the activity level: in normal situations the workload is around 50% of the total capacity. However this workload can exceed 70% in the month preceding important conference deadlines such as the one of SC, GRID, CCGRID, IPDPS, etc. When the workload exceeds 90% (this situation was observed in the Orsay site, the month preceding the SC deadline in 2006), users begin to complain, simply because they are not able to get enough resources or they get them after the deadline.

To understand how users consider Grid'5000 and what should be improved on it, we conducted an audit asking users what they would suggest as improvements of Grid'5000. The inquiry was based on a questionnaire with more than 20 questions. The users are globally happy with the platform. They raised several issues: Grid'5000 is less stable than desired, reproducibility can be improved probably by enhancing the experiment isolation in the nodes and in the intra and inter site networks, the time to prepare an experiment is high compared to the experiment itself, there is a need for more homogeneity in the software and OS settings and more sophisticated network measurement tools. All these comments are very useful and highlight the parts of the system where we should concentrate our efforts. Note that the stability of Grid'5000 is not worse than the one of other production Grids. However since the Grid'5000 users need to reuse the same machines across successive experiments, they have a higher probability to suffer from resource failures.

In addition to its service for research purpose, Grid'5000 is also used for education. A winter school was organized in 2006. 117 participants coming from different communities (computer science, physics, life science) attended courses where they learned how to use Grid'5000, how to run Globus GT 4 on it, how to deploy and run MPI applications on several sites, and how to reconfigure Grid'5000. From the success of this school, we decided to organize schools on an annual basis and to introduce in 2007, the Grid'5000 scientific conference.

Grid'5000 is also used for large scale events. The Grid Plugtests (N-Queens and Flowshop Contests) has used Grid'5000 in a dedicated mode for several days in 2005 and 2006. The purpose of these events is to bring together users of the Proactive middleware and to test the deployment and interoperability of ProActive Grid applications. The Grid Plugtests, which consist of 2 competitions: the N-Queens Contest (find the number of solutions to the N-queens problem, N being as large as possible) and the Flowshop Contest. In 2006, the Grid Plugtests used more than 2600 CPUs during 2 days. For the second consecutive year, Grid'5000 has provided by far the largest number of CPUs among the participating Grids.

4.2. DAS

DAS has had a large impact on computer science research in the Netherlands. It attracted over 200 users and was used so far for more than 30 Ph.D. theses and numerous papers, including many ACM/IEEE journal papers and even a publication in Nature (on astronomy). The DAS systems caused a shift of focus from cluster computing to grid computing and e-Science. Whereas DAS1 was mainly used for doing single-cluster experiments

(e.g., studying different Myrinet protocols [20]), DAS2 is used largely for distributed experiments running on multiple sites.

Perhaps the largest impact of DAS is that it resulted in many new collaborations:

> DAS was a major incentive for the VL-e (Virtual Laboratory for e-Science) project. About one third of this Dutch project consists of computer scientists collaborating through DAS.
> The DAS and Gigaport projects started a collaboration to set up an optical wide-area network for DAS3, as described above. In the StarPlane project, we are investigating how to change the topology of the optical network dynamically, and how to let (e-Science) applications benefit from this flexibility.
> The MultimediaN project is using DAS3 to investigate multimedia content analysis [21] on a grid, including searching in Petabytes of online movie databases and realtime camera data.

VL-e, Gigaport, and MultimediaN each have a running budget between 20 and 40 MEURO. In addition, numerous other grants were obtained from the Netherlands science foundation (NWO) for new research projects on DAS.

The DAS systems also proved to be extremely robust. We have experienced relatively few operational problems. We have used the DAS systems as part of large-scale experiments like GridLab, the Grid Plugtest, demonstrations for the SC conferences, and so on, and over and over again DAS turned out to be of the most reliable components.

5. Toward an international Computer Science Grid

Grid'5000 and DAS3 share many objectives and design considerations. They are complementary platforms. In Grid'5000 reconfiguration mainly concerns the software stack. In DAS3, reconfiguration is focused on the network control. Connecting these two computer science instruments will allow researchers testing Grid interoperability, meta Grid scheduling, security, data transfer, fault tolerance issues, etc. Moreover the connected platforms will allow experiments at a larger scale. Discussion with RENATER may lead to the construction of an equivalent of StarPlane attached to Grid'5000 if experiments on DAS3 demonstrate the benefits of its usage. Thus in the mid term, there is an opportunity to construct and run a full experimental grid equipped with a reconfigurable network.

The connection of Grid'5000 and DAS3 is already on going (Figure 4) . Several people are investigating the technical issues, for example, how to connect the two platforms without comprimising the security of them

In the mid term, a European proposal may be submitted to the Framework Program 7 of the European Community in order to install, run a European Scale Computer Science Grid and investigate Grid and P2P research issues with it. This initiative would provide to the Computer Science researchers involved in the CoreGrid network of excellence a unique, highly reconfigurable and controllable platform for their experiments.

6. Conclusion

Grid'5000 and DAS3 belong to a novel category of research tools for Grid and P2P research: large scale distributed platforms that can be easily controlled, reconfigured

Figure 4. An European Scale Computer Science Grid.

and monitored. The main difference between Grid'5000, DAS3 and the previous real life experimental platforms is their degree of reconfigurability, allowing researchers to reconfigure the software stack (Grid'5000) or the network topology (DAS3) for every experiment.

The construction of a research instrument is a scientific act. Georges Charpak won the Nobel price in 1992 in Physics for his invention and development of particle detectors, in particular the multiwire proportional chamber. Building a Computer Science Grid raises many issues about the quality of the measurements realized using it. Not only measurement tools should be clearly understood and used but the experimental conditions should be imposed and maintained with rigor. Experimental condition injectors such as workload and failure generators for network, processor and disk should be validated. Moreover, since experiments are supposed to test complex systems for long runs, experimental condition injectors may follow sophisticated scenarios.

The construction of such multi-generations large scale instruments is new in computer science and the community is not used to deal with all the administrative, technical and scientific details related to the design, construction, exploitation, maintenance, upgrade and dismantlement of such platforms. Physicists involved in high energy physics and Astrophysicists have a long history of instrument construction behind them. This is a precious source of inspiration for computer scientists.

In addition to be instruments to study Grid research problems, Grid'5000 and DAS3 belong to a novel kind of facilities for computer scientists: platforms with resources opened and shared by a large community of users (typically hundreds). Computer scientists find in these platforms more than just resources they would not be able to access in other circumstances: they find a sophisticated environment involving supporting engineers, specific software, dedicated hardware to ease their experiments and also a social context in which they can share their problems, questions and solutions.

Acknowledgements

Many people are contributing to Grid'5000. Michel Cosnard initiated the project at the national level. Thierry Priol and Brigitte Plateau are respectively the Director of the ACI Grid and the President of the Scientific Committee. We thank the other members of the steering committee: Michel Dayde, Frédéric Desprez, Emmanuel Jeannot, Yvon Jegou,

Stéphane Lanteri, Nouredine Melab, Raymond Namyst, Olivier Richard, Pascale Vicat-Blanc Primet and Dany Vandromme. Pierre Neyron is the head of the Technical committee. We would like to thank the French Ministry of research and the ACI Grid and ACI Data Mass incentives, INRIA, CNRS, RENATER, regional councils of Aquitaine, Bretagne, Ile de France and Provence Alpe Côte d'Azur, Alpes Maritimes General Council and the following Universities: University of Paris Sud, Orsay, University Joseph Fourier, Grenoble, University of Nice-Sophia Antipolis, University of Rennes 1, Institut National Polytechnique de Toulouse / INSA / FERIA / Université Paul Sabatier, Toulouse, University Bordeaux 1, University Lille 1 / GENOPOLE, Ecole Normale Supérieure de Lyon.

Numerous people are involved in the DAS project. Andy Tanenbaum, Bob Hertzberger, and Henk Sips made major contributions in setting up the first DAS systems. We also thank the DAS3 steering group members Lex Wolters, Dick Epema, Cees de Laat, and Frank Seinstra. Kees Verstoep coordinates the system administration work for DAS and also made contributions to this paper. The various DAS systems have been co-funded by the Netherlands Organization for Scientific Research (N.W.O.), the Netherlands National Computing Facilities foundation (N.C.F.), the Virtual Laboratory for e-Science project, the MultimediaN project, and the Gigaport project (which are supported by BSIK grants from the Dutch Ministry of Education, Culture and Science).

References

[1] Franck Cappello, Eddy Caron, Michel Dayde, Frederic Desprez, Emmanuel Jeannot, Yvon Jegou, Stephane Lanteri, Julien Leduc, Nouredine Melab, Guillaume Mornet, Raymond Namyst, Pascale Primet, and Olivier Richard. Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform. In *Grid'2005 Workshop*, Seattle, USA, November 13-14 2005. IEEE/ACM.

[2] Atsuko Takefusa, Satoshi Matsuoka, Kento Aida, Hidemoto Nakada, and Umpei Nagashima. Overview of a performance evaluation system for global computing scheduling algorithms. In *HPDC '99: Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing*, page 11, Washington, DC, USA, 1999. IEEE Computer Society.

[3] Henri Casanova, Arnaud Legrand, and Loris Marchal. Scheduling distributed applications: the simgrid simulation framework. In *Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, may 2003.

[4] C. Dumitrescu and I. Foster. Gangsim: A simulator for grid scheduling studies. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05), Cardiff, UK*, may 2005.

[5] Xin Liu, Huaxia Xia, and Andrew Chien. Validating and scaling the microgrid: A scientific instrument for grid dynamics. *The Journal of Grid Computing*, Volume 2(2):141 – 161, 2004.

[6] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):00–00, July 2003.

[7] Y. Georgiou, O. Richard, P. Neyron, G. Huard, and C. Martin. A batch scheduler with high level components. In *Proceedings of CCGRID'2005*. IEEE Computer Society, 2005.

[8] P. Augerat, C. Martin, and B. Stein. Scalable monitoring and configuration tools for grids and clusters. In *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*. IEEE Computer Society, 2002.

[9] Yiannis Georgiou, Julien Leduc, Brice Videau, Johann Peyrard, and Olivier Richard. A tool for environment deployment in clusters and light grids. In *Second Workshop on System Management Tools for Large-Scale Parallel Systems (SMTPS'06)*, Rhodes Island, Greece, April 2006.

[10] A. Iosup, C. Dumitrescu, D.H.J. Epema, H. Li, and L. Wolters. How are real grids used? the analysis of four grid traces and its implications. In *7th IEEE/ACM Int'l Conference on Grid Computing (Grid2006)*, sept 2006.

[11] Gabriel Antoniu, Loïc Cudennec, Mike Duigou, and Mathieu Jan. Performance scalability of the JXTA P2P framework. In *Proc. 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Long Beach, CA, USA, March 2007.

[12] M. Mezmaz, N. Melab, and E-G. Talbi. A grid-enabled Branch and Bound Algorithm for Solving Challenging Combinatorial Optimization Problems. In *In Proc. of* 21^{th} *IEEE Intl. Parallel and Distributed Processing Symp.*, Long Beach, California, 26–30 Mar. 2007.

[13] Stéphane Genaud, Marc Grunberg, and Catherine Mongenet. Experiments in running a scientific mpi application on grid'5000. In *IEEE International Workshop on Grid Computing (HPGC), IPDPS 2007*. IEEE Society Press, march 2007.

[14] S. F. Portegies Zwart, H. Baumgardt, P. Hut, J. Makino, and S. L. W. McMillan. Formation of massive black holes through runaway collisions in dense young star clusters. *Nature*, 428:724–726, April 2004.

[15] Thilo Kielmann, Philip Hatcher, Luc Bougé, and Henri E. Bal. Enabling Java for High-Performance Computing: Exploiting Distributed Shared Memory and Remote Method Invocation. *Communications of the ACM*, 44(10):110–117, October 2001.

[16] Tom DeFanti, Cees de Laat, Joe Mambretti, Kees Neggers, and Bill St. Arnaud. TransLight: a global-scale LambdaGrid for e-science. *Communications of the ACM*, 46(11):34–41, November 2003.

[17] John W. Romein and Henri E. Bal. Solving the Game of Awari using Parallel Retrograde Analysis,. *IEEE Computer*, 38(10):26–33, October 2003.

[18] Gosia Wrzesinska, Jason Maassen, and Henri E. Bal. Self-adaptive applications on the grid. In *Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, March 2007.

[19] F.J. Seinstra, J.M. Geusebroek, D. Koelma, C.G.M. Snoek, M. Worring, and A.W.M. Smeulders. High-Performance Distributed Image and Video Content Analysis with Parallel-Horus. *IEEE Multimedia*, 2007. in press.

[20] K. Verstoep, R.A.F. Bhoedjang, T. Ruhl, H.E. Bal, and R.F.H. Hofman. Cluster Communication Protocols for Parallel-Programming Systems. *ACM Transactions on Computer Systems*, 22(3):281–325, August 2004.

[21] F.J. Seinstra, D. Koelma, and A.D. Bagdanov. Finite State Machine-Based Optimization of Data Parallel Regular Domain Problems Applied in Low-Level Image Processing. *IEEE Transactions on Parallel and Distributed Systems*, 15(10):865–877, October 2004.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

An e-marketplace model for logistics services based on Grid technology

Maria Carmen INCUTTI [a], Francesco MARI [b]
[a] *DEIS, Università degli Studi della Calabria, Via P. Bucci 41C, Rende (CS) 87036, Italy*
[b] *CESIC-NEC, c/o Università degli Studi della Calabria, Via P. Bucci 22B, Rende (CS), 87036, Italy*

Abstract. New technologies are like accelerators for the economic growth. According with this trend the way logistics is managed is changing; there is a move to: faster, more reliable logistics networks; global purchasing for cost reduction; high-speed logistics services through information systems, e-business. E-commerce is here to stay, to grow and to transform the way logistics business is conducted. E-marketplaces are a logical extension of the ability of e-business companies to trade goods and services online. The goal of this chapter is to understand the possibilities and needs using electronic marketplaces and Grid technology for the procurement of logistics services. A new e-commerce model is suggested. The model aim to integrate logistics service providers to companies that need to have their products distributed. To solve computation intensive optimisation tasks, to manage distributed data and to integrate back-office and marketplace applications, a Grid based marketplace implementation model is outlined.

Keywords. Logistics, e-marketplace, grid technology.

Introduction

Increasingly competitive markets are making it imperative to manage logistics systems more and more efficiently. Logistics [1] is key to the modern economy: almost every organization faces the problem of getting the right materials to the right place at the right time. Logistics Management is a business concept related with the opportunity/necessity to plan, implement and control the efficiency of the supply chain. While in the early 1950s was crucial the possibility of strategically manage and control goods, services and related information in a integrated view of all the business activities, nowadays an efficient logistics management is necessary to deal with the increasing complexity of globalized supply chain.

Today, staying ahead of the competition means making the most of limited resources and planning effectively to identify and select new opportunities; the way to do this is optimizing supply chain utilizing new technology solutions.

In the context of today's business world, integration between traditional supply chain optimization and the new information technology is becoming increasingly important to companies in their attempt to achieve competitive advantage, to meet the

constantly changing demands placed on organisations to excel in their sector. The widespread development of information and communication technologies have completely changed the way logistics is managed, with remarkable benefits both in terms of cost reduction and service level improvement. The new logistics management paradigm consists in exploiting up-to-date parallel and distributing computing resources in order to process the huge amount of information required to synchronize customer orders, production schedule and distribution plans along the supply chains. Who is hesitant to jump into a new technology may be missing out on competitive opportunities.

When most people think about technology, they tend to reflect only upon the application layer, but it is necessary to think also on the data and the whole on-demand approach. It regards work flow and data management and integration of information, i.e. the ability to share data, connect data, and make decisions regarding supply chain.

It is essential to identify and manage rules in order to get the right information available for people who need it. Technology advances improve supply chain visibility in a variety of ways. The key is how quickly setting the information up and how efficiently using it.

Complete implementation of these systems allows us to talk of real time across different organizations, of course some elements may be better optimized than others, but this gives us a remarkable ability to make the model work.

Knowing the who, what, where and when of buying can give to a business entity a better opportunity to work with its supplier to put in place supplier/customer relationships on their commercial end to smooth the supply chain. Even if it is not possible to work directly with the suppliers, it is necessary to make decisions regarding supply chain with third-party logistics providers (3PLs) and technology is capable of offering efficient solutions.

Today a large part of supply chains are managed across the Internet, unexpectedly this management still contains a lot of inefficiencies. The Internet had a tremendous impact on the field of supply chain management, high number of companies have used the Internet successfully to lower costs, improve response times and add value to their businesses. Nowadays the Internet, even if potentially able to create business opportunity, is not sufficient, other technological solutions are necessary. Solutions may be researched in two directions: Web services [2] and Grid computing [3].

The first direction is represented by the definition of efficient Web services capable not only of simplifying information exchange and business processes within the enterprise and between supply chain partners, but also of supporting business decisions. The benefits that may be derived in logistics service organisations through the use of Web services technology are important. As explained above, collaboration is extremely important for business companies, thus the implementation of distributed computing technology is an essential element of the business strategy.

Web services emerged as the potential integration technology of the future, can be characterised as self-contained, self-describing, modular applications that clients can publish, locate, and dynamically invoke across the Web. Web services enable the development of distributed enterprise applications, improving data exchange with business partners and lowering costs of integration. This technology, which takes advantage of the ubiquitous nature of the Internet, permits an important integration in the context of logistics services and in particular for the development of robust, useful and cost effective solutions for on-line logistics services.

The second direction is represented by an efficient use of the distributed resources of a grid computing environment. Grid and peer-to-peer technology may be successful when used to solve logistics and transportation optimization problems. The recent technological advances in grid environment get us think that in the near future many conventional industrial applications will be interested in the distributed computing. Companies face every day computationally intensive problems characterized by a large set of normative, functional and operational constraints.

A medium/large company operating in the logistics sector (or widely interacting with it) typically has thousands of customers to service every day, each one has to be geo-coded on a detailed road map and serviced within a specific time window. For this reason, the company may use an heterogeneous fleet of vehicles that has to be routed and scheduled.

The optimization and planning of complex real-world logistics problems requires a huge computational effort, both in terms of CPU and memory structures in order to code the customers of the day on the road map, compute a huge time/distance matrix among customers basing on distributed data and define an effective routing plan.

Advanced optimization algorithms to solve these problems often exist but the available computing power is not sufficient to guarantee acceptable response times, so heuristic solutions are used. Even if today effective methodologies, generally inspired by natural phenomena, have been developed to solve real industrial problems, they can not be used, because their computing cost is too high and low quality solutions are often chosen.

Grid computing gives the possibility to solve real industrial problems by advanced optimization algorithms reducing dramatically computing time, and permits to effective and quickly process great quantity of distributed data.

Nowadays, the literature is full of parallel algorithms for a large set of optimization problems, but very few of them are grid-enabled algorithms. Parallel computing provides substitute design alternatives for solution algorithms and offers also the opportunity to obtain performance benefits in both computational times and solution quality. Very often this is not enough: complex problems, which can only be solved in non-polynomial time and need huge distributed data sets, arise in most research fields and are becoming common in many research areas (industrial environments, economy, telecommunications, etc.). In other words many optimization problems cannot be solved to optimality within reasonable amounts of time with parallel computational resources.

In order to find good solutions to these computationally demanding problems, grid-enabled algorithms need to be developed. Grid technology gives the possibility to solve optimization problems, otherwise out of the scope of researchers dealing with parallel algorithms.

Recently, some papers on optimization algorithms on grid have been published. In particular, in literature are available some recent papers about branch-and-bound algorithms on Grids. The algorithms, that implement the branch-and-bound technique, search the solutions space following a tree enumeration. The computations along the sub-trees can be realized almost independently; this is the reason why researchers consider these algorithms suitable for parallel and grid computing.

Iamnitchi and Foster (2000) [4] have proposed a fully distributed branch-and-bound algorithm with fault tolerance mechanisms. Goux et al. (2001) [5] have proposed a centralized framework based on a master–worker model. In 2002 Anstreicher et al. [6] improved the previous model adding a load balance strategy between workers. In 2003

Aida et al. [7] proposed a distributed branch-and-bound algorithm based on a two-level hierarchical master–worker algorithm. In 2006 Drummond et al. [8] proposed a fully distributed branch-and-bound algorithm that includes load balance and fault tolerance, reducing also the communication in low speed links. In 2007 Bendjoudi et al. [9] proposed a P2P-based parallelization of the branch-and-bound algorithm for the computational Grid.

For our knowledge, no grid-enabled solution algorithm is currently used to solve real-world industrial logistics problems, even if the advantages obtainable by using for addressing them such technology are evident. A grid is able to offer a good solution for the high computing requirements of a lot of logistics applications. Enabled by grid technological progress, real industrial applications will possibly consider very large instances, real road maps and a large set of operational constraints. It could also provide much better solution and more detailed simulation. The possibility to realize fast response tools basing on grid technology will give the opportunity of integrate modules for high level logistics solutions to industrial Enterprise Resource Planning systems (ERP). As a result the grid will be an important instrument to increase company competitiveness.

The current trend of the ERP company is to try to provide value-added integrated solutions; one of the most important solution is the optimization of internal and external logistics functions. Effective logistics optimization, with special reference to transportation costs, is highly computationally intensive. Moreover, companies characterized by a great number of customers need to populate data structure containing million of shortest distances on detailed road maps; this induces severe storage requirements. Generally, the problems are characterized, also, by high time constraints for getting a solution. As a result real problems are very difficult to solve. The grid should provide very flexible solutions consistent with the industry real needs as required from the companies.

The grid technology could: dramatically reduce computing times; increase a central control on distributed activities and their costs; support the definition of distributed scenario; support a good cost analysis based on an efficient data management and consequently reduce the overall cost. It could also provide instruments for efficient price policies, working load balancing of the drivers and depots, fleet vehicle optimization.

The idea is to put together web services and grid into logistics management; in particular we propose an e-marketplace model for logistics services based on web services and grid technology.

The chapter is structured as follows. In Section 2, the relationship between web services and grid computing is illustrated. In Section 3 a general description of the e-marketplaces for logistics services together with a literature review is given. In Section 4, the advantages of basing an e-marketplace system on grid technology are shown. Our suggested e-marketplace model is presented in Section 5. Finally, the conclusions of our research activity and an overview of future work are given.

1. Web services and grid computing

Grid computing has emerged as a global platform to support organisations for coordinated sharing of distributed data, applications, and processes. While the promise of the Grid was fantastic, still this promise has not been fully realized. On the other

hand, emerging Web Services have transformed the Web into a dynamic and powerful content. Web services-based applications may be the solution to allow the Grid to realize its potentialities.

The original idea behind the Grid was to solve computation intensive tasks by connecting the processing power of distributed resources over the network. Grid computing, devised in research organizations to support compute-intensive scientific applications and to share massive research databases, is a form of distributed computing in which the use of disparate resources often spread across different physical locations and administrative domains. Recently Grid computing has began to move out from the laboratories towards more general applications: enterprises of all types are beginning to recognize the grid technology as a foundation for flexible management and use of their internal IT resources, enabling them to better meet business objectives such as minimized cost and increased agility and collaboration. Grid technology gives the possibility to different scientific and business partners to collaborate: more and more emphasis has been placed recently on this opportunity. In these cases the problem is not only to run large programs on distributed resources but also enable business partners to use hardware or software services and databases. Grid computing is thus the foundation for collaborative high-performance computing and data sharing, for the adaptive enterprise, and for the vision of utility computing, in which computing resources will become pay-per-use commodities.

A rapidly emerging field in distributed computing is Web services. Web services are distributed software components that provide information to applications through an application-oriented interface. They include messaging protocols, standard interfaces and directory services, as well as security layers, for efficient and effective business application integration. Typical application areas are business-to-business integration, business process management and design collaboration. The information is structured using *eXtensible Markup Language* (XML), a markup language for formatting and exchanging structured data; this way the information can be easily processed.

To completely describe Web services it is necessary to illustrate other specifications, in particular SOAP and WSDL.

SOAP is the acronym of Simple Object Access Protocol that is its originally meaning, today it has a different meaning but it maintains its original name; actually it is an XML-based protocol for specifying envelope information, contents and processing information for a message. Although a Web service can support any communication protocol, the most common is SOAP over either HTTP or HTTPS. This contributes to the appeal of Web services; the main advantage of this approach is its ubiquity which makes interoperability very simple. The ease with which Web services can be implemented and accessed from any platform has led to their rapid adoption by system management bodies that provide common manageability interfaces to disparate resources.

WSDL is the acronym of Web Services Description Language; it is an XML-based language used to describe all the properties of a Web service, in particular attributes and interfaces. A WSDL document can be read by a potential client to learn about the service. Web services give details of their functions and interfaces, but they keep private their implementation; thus, a client and a service that support common communication protocols can interact without taking into account the programming languages in which they are written or the platforms on which they run. These web

services' characteristics make them particularly appropriate to a distributed environment.

Staab et al. (2003) [10] outline five issues that Web services need to address: *efficiency, expressiveness, security, reliability and manageability*.

The concept of *efficiency* means that the execution of Web services must be efficient to ensure scalability to enterprise application integration.

Expressiveness means that the complex enterprise business processes required for integration purposes necessitate expressive concept modelling.

The concept of *security* means that data transfer across and between enterprises must be secured in order to prevent attack and loss of data integrity.

Reliability means that communication using Web services must be reliable so that messages are sent once to ensure dependable interactions between the involved parties.

Manageability means that the Web service process for e-business automation must be manageable so that organisations can monitor usage and delivery and also enable easy update to Web services interfaces due to frequent enhancements.

Grid computing and Web services have different backgrounds but a similar target: a fully integrated computer network that performs distributed computation with the best matched device. Grid computing has also started to use Web services to define standard interfaces for business services on demand. Service orientation of the Grid makes it a promising platform for dynamic development and integration of service-oriented applications. As a result these two technologies work together to providing a powerful computing platform to the business world.

The Web services technology is non-proprietary, so solutions implemented using these technologies are relatively inexpensive. Moreover, the solutions are flexible due to the platform-independence of the technology, and the ubiquity of the Internet.

The convergence of Grid computing and Web services is also embodied by the specification of Open Grid Services Architecture (OGSA) [11]. The OGSA Architecture document defines a service oriented Grid architecture based on Web services standards and protocols. It describes, at a high level, the required capabilities of the service-oriented grid (job execution, resource management, data management, and security) as well as interfaces and standards to ensure interoperable implementations and a core set of services.

A Grid service is a special Web service that provides a set of well-defined interfaces and that follows specific conventions in order to address a core set of services. Since every Grid service is a Web service, it has to implement several interfaces. The Open Grid Services Infrastructure specification [12] describes these interfaces using an extended version of WSDL.

Since Web services are becoming more and more common in business applications, a Grid architecture is the natural model to adopt in a business environment. Therefore, the more the OGSA specifications become mature, the more intensive become the interests of large industry players towards Grid-based solutions.

This chapter proposes a Grid services based model for an e-marketplace for the procurement of logistics services. E-marketplaces are based on initiatives for buying or selling products between customers and merchants or initiatives for supplying products among merchants, in the supply chain. They help trading partners to minimize their costs and increase their profit. In spite of the growing number of recent contributions in the area of e-commerce, it is necessary to face one serious challenge: the lack of appropriate support for managing e-marketplace initiatives.

The online optimization that e-marketplaces can offer requires large amount of computations and does not permit any delay. Moreover, the information that electronic marketplaces need are subjected to changes and evolutions. For these reasons, Grid technology can be viewed as a powerful solution.

This chapter aims to carry this consideration one step forward. Therefore, in the following sections a Grid services enabled e-marketplace based on Grid technology is outlined.

2. E-Marketplaces for the management of logistics services

Electronic marketplaces, capable of acting as important facilitators of business activities, seem to be potentially able to turn upside down the logistics management.

An e-marketplace is a virtual online market where organizations conduct business-to-business (B2B) e-commerce over the Internet. Services offered by e-marketplaces include electronic catalogues for online purchasing of goods and services, business directory listings and online auctions. E-Marketplaces are at the core of future e-commerce growth. It is predicted that e-marketplaces will comprise 75% of worldwide online B2B trade, and they are currently growing at a faster rate than private e-procurement and e-distribution solutions.

E-Marketplaces differ from private e-procurement (buyer-centric) and e-distribution (seller-centric) solutions in that they are usually initiated by a third party and serve multiple interests on both the buyer and seller sides. There are many types of electronic market based on a range of business models. They can be operated by an independent third party, or be run by some form of industry consortium that has been set up to serve a particular sector of business. Electronic marketplaces can be divided into categories basing on the way in which they are operated, in particular we distinguish between horizontal and vertical marketplaces. Horizontal marketplaces are category-specific, usually specializing in offering indirect goods. Companies from different industries can leverage consortium buying and pre-negotiated contracts to reduce prices and also increase the efficiency of non-strategic procurement processes. Vertical marketplaces are industry-specific, offer strategic products and are more focused on adding value by building communities and firm relationships between trading partners. Moreover, e-marketplaces can be either public or private. Public marketplaces are open to all companies, usually enabling participation after a credit check and registration. Private marketplaces are closed, usually set up for a group of co-operating buyers and exclusively invited suppliers. In addition, marketplaces can be divided into supplier-oriented, buyer-oriented and independent marketplaces.

A supplier-oriented e-marketplace is set-up and operated by a number of suppliers who are seeking an efficient sales channel via the Internet to a large number of buyers.

A buyer-oriented marketplace is normally run by a consortium of buyers in order to establish an efficient purchasing environment. The intention is to help buyers to lower their costs and improve the communication with suppliers. The e-marketplace is open to existing suppliers who are encouraged to place their catalogues online so that they can be accessed by all members of the consortium.

An independent e-marketplace is operated by a third party who is neither a buyer nor a seller. They are usually public marketplaces open to all buyers and sellers in a particular industry or region. Typically some form of payment is required to participate.

E-marketplaces may also offer different extents of trading mechanisms and value-added services.

In the last few years an increasing number of manufacturing companies has understood the enormous advantages of the strict collaboration among customers, suppliers and logistics operators. This approach, known as "collaborative logistics", aims at the identification and reduction of the hidden costs that the logistics subjects have to pay without having full control on them. For example, in the UE market the asset repositioning cost (the cost to reposition the inactive resources in order to make them ready to be used by the successive customer) has reached the value of the 18% of the total distribution cost (that means 165 billion euros).

We started from the idea to project an e-marketplace of the logistics services, that is based on the collaborative logistics approach and capable of well coordinating demands and offers of logistics services and, therefore, of increasing the resources usage level.

We have analyzed other different e-marketplaces, which are dedicated to the transportation of goods [13][14]. Those systems give the opportunity to distribution companies of offering vehicles for goods distribution and to other companies to publish product transportation demands. The mission of both systems is to match the transportation requests with the transportation offers but the deal is reached by the actors by themselves (blackboard approach).

A more significant approach is the exploitation of the operations research to effectively support the e-marketplace system, making it more powerful, effective and efficient. Our idea is to propose in this chapter an e-marketplace framework based on a combinatorial auction system.

Combinatorial auctions are those in which the auctioneer places a set of heterogeneous items out to bid simultaneously and in which bidders can submit multiple bids for combinations or bundles of these items. Further, bids can be structured and described with sophisticated logical relationships so that bidders can make conditional bids that properly express their preferences for different collections of items. In recent years, combinatorial auctions have proved to be a very effective and economically efficient price discovery mechanism and they have attracted a significant number of operations researchers and economists. Combinatorial auctions based systems have been used with benefits in a variety of industrial application areas, but of primary interest to our research is the procurement of contracts for freight transportation services.

In freight transportation service procurement, shippers (typically large manufacturing companies or retailers) put service contracts on sale to carriers (trucking companies, for example) based on negotiated transportation rates and service level agreements. These contracts are for the provisions of transportation services between specific origin–destination pairs (*lanes*). It is worth to notice that a very important factor contributing to define carrier's transportation costs (and therefore its rates) is the empty movement cost associated with vehicles move. As a result, carriers place a higher value on groups of lanes than on the sum of individual lanes if they can reduce their empty movement costs by combining these lanes into a single route. For these reasons, recently shippers have begun to sell contracts to carriers using combinatorial auctions instead of requesting quotes for each lane or small groups of lanes separately. This practice has reportedly resulted in significant benefits to shippers.

However, it is well known that the bid valuation and construction problem for carriers facing combinatorial auctions for the procurement of freight transportation

contracts is very difficult and involves the computation of a number of NP-hard sub problems. In literature, several recent published papers point out that it is computationally difficult for the bidders to valuate the values or costs associated to each possible item to bid for [15][16]. Contrary to the assumption in classic auction theory that bidders know these information a priori, bidders in a combinatorial auction have to devote resources to discover their true values and to develop bids.

Another important problem to solve in a combinatorial auction is the so called Winner Determination Problem (WDP). A WDP is a particular optimization problem where the objective is to find the winner (or winners) of combinatorial auctions in order to either maximize an objective function which represents the gain for the sellers or to minimize a function which represents the costs that the customers pay to obtain the services on sale on the marketplace. It is a well-known operations research problem, that belongs to the NP-Hard class of problems. As one can easily understand, how to address the WDP in different combinatorial auctions is the real basis of the e-markeplace system that we are proposing and that will be detailed in the next sections. In the following we give some literature review about how those optimization problems have been addressed.

In recent years, the design and use of combinational auctions across many applications have received an high interest by operations researchers and economists [17]. These auctions are suitable for situations in which complementarities and/or substitution effects exist among different combinations of items and in which bidders have complicated preferences for group of items rather than for individual items. One of these situations is the procurement of trucking services.

The costs related to the vehicles repositioning from the destination of one load to the origin of a subsequent load represent an high percentage of logistic costs in trucking operations. Traffic lane operations exhibit strong interdependencies, that is, the cost of serving one lane often is strongly connected on the opportunity to serve other lanes. Specifically, the cost of serving together a set of lanes may be less than, equal to or greater than the cost of serving them by separate routes. For example, a lane from Rome to Milan is not related to a lane from Naples to Palermo; as a result, it does not make a difference if they are contracted to two carriers separately or to a single carrier. However, a lane from Naples to Rome is complementary to a lane on the return trip from Rome to Naples which makes the combined operations more cost-effective. As a result, carriers have complicated synergies over combinations of service contracts. This property makes combinatorial auctions a particularly suitable resource allocation mechanism for trucking service procurement and give to carriers a good opportunity to reduce their procurement costs [18].

Recently, large shippers have started to experiment with combinatorial auctions with the intention to give carriers flexibility in bidding and eventually to reduce their procurement costs. For our knowledge, the first company to use this mechanism to procure truckload trucking services has been Sears Logistics Services. Ledyard, Olson, Porter, Swanson and Torma [19] reported that in 1995 the company conducted a multi-round combinatorial auction to sell contracts serving over eight hundred lanes and involving a cost of nearly two hundred million dollars per year. The result was that Sears Logistics Services reported a 13% savings that reduced its transportation procurement cost by $25 million per year. Another big company, The Home Depot, experimented in 2000 a one-shot, sealed-bid combinatorial auction to procure services for its truckload shipments [20][21]. In that application, the lanes that were auctioned-off accounted for about 52000 moves, approximately one fourth of all the inbound moves

to stores within Home Depot's network. Over 110 carriers were invited to participate and a majority of them submitted bids. The authors reported that experiment as a big success even if savings figures were not disclosed. Additional applications of combinatorial auctions in the procurement of freight transportation contracts have been presented by Caplice and Sheffi in 2003 [22], by Sheffy in 2004 [23] and by Elmaghraby and Keskinocak in 2000 and 2003 [20][21] .

It is important to mention that several difficult problems that have to be addressed are associated with combinatorial auctions. The problem of assigning winning, non-conflicting bids to bidders with maximal benefits (Winner Determination Problem, WDP), has received the most attention in the research related to combinatorial auctions. It can be formulated as a variant of either the set covering problem or the set partitioning problem or the set packing problem depending on whether it is a forward auction (revenue maximization) or reverse auction (cost minimization) and depending upon auctioneers constraints. In all cases these are NP-Hard problems. Both exact and approximation algorithms have been studied in the past for the WDP. A detailed review of formulations and algorithms for the winner determination problem has been published by De Vries and Vohra (2003) [17]. While some variants of the Winner Determination Problem are well solved, when additional side constraints such as those identified by Caplice and Sheffi (2003) [22] are added, the problems become even more complex. In response to these complexities, recent research has focused on conducting simpler unit auctions for the trucking application in which the shippers predefine mutually exclusive and collectively exhaustive bid sets a priori or by developing approximation algorithms tailored to the more complex formulation of the problem [24][25].

The auction mechanism design, the question of how to design auctions in order to induce bidders to bid their true interests and achieve economic efficiency, has been a topic of interest in auction theory for many years. An important question in the design of combinatorial auctions is the bidding language, or syntax that is specified by auctioneers for bidders to express their logical preferences over combinations of bid items. A ''good'' bidding language should be both expressive and simple so that bidders are able to state their synergies on their desired combinations of items and they can use it easily. Nisan (2000) [26] formally introduced eight combinational bids including or extending three basic types. These are: *atomic bids* in which a bundle of items are treated as a single indivisible bid; *OR bids* which are collections of atomic bids in which the bidder will serve any number of disjoint atomic bids for the sum of their respective prices; and, *XOR bids* in which the bidder will serve at most one out of a set of atomic bids at the specified price. Sandholm (2002) [27] identified how combination bids, called *OR-of-XOR bids* could be used to represent all necessary relationships. Abrache et al. (2004) [28] discussed the limitation of Nisans' language and proposed a two-level bidding framework and analyzed its impact on the formulation of the Winner Determination Problem. In addition, a bidding language can be designed to induce bidders make a reasonable number of bids so as to reduce the complexity of combinatorial auctions. Rothkopf et al. (1998) [29] discussed the restricted structures of bids under different scenarios in combinatorial auctions so that the number of bids generated can be tractable. Most recently, Day and Raghavan (2003) [30] presented a method for generating tractable bid sets by imposing certain restrictions known as matrix bids with order.

A new problem generated by combinational auctions is the threshold problem. It is a variant of the free-rider problem and occurs when small bidders fail to coordinate

their bids to compete effectively with large but inefficient bids. This occurs because each of these small bidders intends to bid less anticipating that other bidders will bid high so that they can be a ''*free rider*'' (Cramton, 2002) [31]. Banks, Ledyard, Porter (1989) [32], Bykowsky, Cull, Ledyard (2000) [33] and DeMartini, Kwasnica, Ledyard and Porter (1999) [34] proposed new designs to avoid this problem in multi-item auctions. In addition, Parkes and Ungar (2000) [35][36] discussed the design of iterative auctions to reduce the complexity of the Winner Determination Problem and analyzed their equilibrium conditions. All of these research problems are from an auctioneer's perspective. A classic assumption in auction theory is that bidders know their true values or costs for the items they want to procure prior to the auction or that value or cost is common across all participants but unknown due to missing information [37]. However, this assumption might not be true in a combinatorial auction, especially when a large number of combinational opportunities exist and when bidders have hard local optimization problems to solve. Larson and Sandholm (2001) [38] showed that the Generalized Vickrey Auction protocol loses its dominant strategy property when bidding agents are unrestricted but have limited computational resources. Conen and Sandholm (2001) [39] observed the existence of an exponential number of bundles that bidders may need to compute and therefore proposed an auctioneer agent that uses a topological preference structure to request only necessary information, and as a result, reduces the number of valuation problems that bidders need to solve. Further, they presented a method to make their design incentive compatible so that bidders need only to compute their own preferences and not those of competing bidders. Jones and Koehler (2002) [40] developed a framework within which instead of specifying bids completely, bidders present the auctioneer with a set of rules or constraints that the solution of the winner determination problem must satisfy. Parkes (2000) [15] acknowledged that market design alone cannot fully simplify the bidder's valuation problem, but argued that a well-designed auction could improve the quality of the bidder's decisions. Further, Parkes et al. (1999) [41] introduced a bounded rational compatible auction in which a bidding agent makes bidding decisions based only on approximate information about the value of a good, that is, lower and upper bounds on its true value. The bid construction problem in freight transportation contract procurement may be even harder than that solved in other combinatorial auctions. In the trucking industry, carriers not only need to consider the economies of scope exhibited in delivery routes from new contracts, they also have to find an efficient way to integrate these new contracts with their pre-committed contracts. This, normally modelled as a vehicle routing problem, is itself NP-Hard in most cases as its solution typically requires the solution of variants of multiple travelling salesman problems. The solution of this problem provides a carrier's true costs for serving a specific set of new contracts only. Research on solving vehicle routing problems is common. Extensive reviews of the basic vehicle routing problem, time constrained routing and scheduling and dynamic and stochastic routing and scheduling can be found in Fisher (1995) [42], Desrosiers (1995) [43] and Powell, Jaillet and Odoni(1995) [44] respectively.

Combinatorial auctions have been the topic of active research during recent years. However, there have been few attempts to address the bid construction problem from the perspective of a bidder. Song and Regan [16] propose a formulation of the problem to address to help a bidder to construct its optimal bids and propose an approximation method to address it. Of particular interest are the following questions: How should bidders determine their true values or costs for any bundle of bid items? What is the optimal way to structure bids for different combinations of items? These questions are

not easy to answer and need adequate computational resources to be well addressed even for simple cases.

3. E-marketplaces based on Grid technology

The potential advantages to be gained by joining an e-marketplace may be important, of course there are general business benefits and specific business benefits, these ones vary among industries and businesses, and between sellers and buyers. As general business advantages we could enumerate: the possibility of new trading partnership, the transparency and the reduction of business time. The virtual online markets give great opportunity for suppliers and buyers to establish new trading partnership either within their supply chain or across supply chains. They can also provide great transparency in the purchasing process since prices, availability and stock levels of the goods or services are all available in an open environment. Lastly, time constraints and problems with different office hours for international trade may be easily removed.

While these are the general business benefits, it is necessary to distinguish between other important advantages of buyers and sellers of the e-marketplaces.

The virtual online markets offer to the buyers the opportunity to have always updated information, to efficiently compare products and their cost, to provide a high level of trust. Updated information on price and availability make it easier to secure the best deal to the buyers. E-marketplaces offer a convenient way to compare prices and products from a single source rather than spending time contracting each individual supplier. This way the buyers can save their money and business time getting the lowest price on what they need to buy. E-marketplaces also provide a high level of trust for the buyers as they deal exclusively with suppliers that are perfectly known, because member of the virtual market.

The virtual online markets offer to the sellers the opportunity to significantly increase the number of customers and reduce their costs. E-marketplaces provide additional sales channel for their products or services, in particular international e-marketplaces can provide chances for overseas sales which can represent good business opportunities. Lastly, the electronic markets offer reduced marketing costs when compared to other sales channels.

Summarizing, the main benefits for a seller of becoming involved in an e-marketplace are reduced sales costs, greater decision flexibility, saved time and better collaboration with other partners.

In addition to these important advantages, an e-marketplace model based on Grid technology and Grid services offers other remarkable advantages.

The first one is the possibility of integrating existing value-added services. Actually e-marketplaces offer only a restricted number of functionality because of the difficulties in integrating marketplace activities with ERP systems and legacy back-office applications. This integration is a very complex challenge but is necessary in order to achieve an effective exploitation of the potentiality of the system. An e-marketplace model based on Grid and Web services concepts offers solutions for this problem. As noticed above, interoperability is provided by Web services standards. As Grid services are particular Web services they are able to communicate by exchanging standard SOAP messages despite any differences in hardware platforms, operating systems or programming languages applied. Therefore, if marketplace applications as

well as back-office applications are implemented as Grid services they are capable of communicating.

The second important advantage offered by an e-marketplace model based on Grid technology is the possibility to distribute the amount of calculation among a set of distributed computing resources. An e-marketplace application may require large computational power. In particular a great number of efficient logistics optimization algorithms ask for massive computing resources. In order to give quick and efficient solutions an e-marketplaces may use distributed computing resources.

The third important advantage offered by an e-marketplace model based on Grid technology is the possibility of an efficient and effective management of distributed data. The nature of the modern logistics management involves complex information flows between multiple organizations. To develop good solutions it is necessary to implement automated information processing between organizations. The data that are distributed throughout the supply chain has to be carefully analyzed and then used to realize strategic goals. Each organization has to integrate with external partners across the supply chain so that the data visibility is apparent and all the participants of the integration take advantages from it in order to lower the costs and/or increase consumer confidence in a brand or product. The long term strategic benefits to the companies are tangible: effective integration of systems across diverse locations and multiple organizations is essential to manage the return of consumer products; an efficient and effective transfer of data between organizations is the only way to implement this integration. Additionally, organizations can utilise the technology internally to improve reporting functions. Strategically, this enables the production of quality management reports that aid the decision-making process in service organizations.

An e-marketplace model based on Grid technology offers the possibility to use this important resource giving effective instruments to query distributed database across different organization in order to provide access to real-time data for more accurate strategic decision-making processes.

The Grid is a global platform to support organizations for coordinated sharing of distributed data, applications, and processes. It is clear that the advantages offered by E-marketplaces based on Grid technology are remarkable. However, if marketplace participants (buyers and sellers) should reengineer their existing applications (or develop completely new versions of them based on Grid services) these advantages could probably remain only a promise. The reengineering process is time consuming and very expensive, so probably a lot of companies could decide not to change their traditional habits. Clearly, if the reengineering process can be not so difficult, it is imaginable that the marketplace participants could easily try to experiment it.

Delaitre et al. [45][46][47] offers a relatively easy solution for this: the Grid Execution Management for Legacy Code Architecture (GEMLCA). GEMLCA is a general architecture to set up legacy code programs as Grid services without code re-engineering. It is a Grid service that offers a number of interfaces to submit and check the status of computational jobs, and give the results back. As any other Grid service, GEMLCA has an interface described in WSDL that can be invoked by any Grid service client to use its functionality through SOAP. The idea is to transfer legacy applications into Grid services only by installing a software solution, so that the existing applications do not require any modification in the original code. Therefore, marketplaces could immediately utilize the advantages of a Grid service-based model. If marketplace's operators, buyers and sellers install GEMLCA, marketplaces can immediately exploit the benefits of a Grid service-based model. This architecture,

giving the possibility to transform existing applications into Grid services without code reengineering, offers a easy way to take all the advantages of the Grid technology.

4. An e-marketplace model for selling transportation contracts using e-auctions

Logistics is an important component of the business management. It involves the transfer of goods or services from the point of manufacture to the point of consumption in order to meet the market demand [48]. As noticed above, currently the main part of the logistics management is provided across the Internet. While in the past this represented for logistics activities a clear improvement because telecommunication through computers facilitated a new relationship between businesses, today this is not enough and new logistics solutions have to be found. Internet, for example, does not enable an optimization that would include logistical costs. Promising tools for logistics services management seem to be the e-marketplaces. Today the existing e-marketplaces sell goods; usually they do not offer logistical solutions at all or offer a single solution or very few possibilities in different time-cost ranges. As explained above a company that join a marketplace has to face technical problems in order to integrate its legacy systems with the marketplace solutions. This integration task is often difficult and time consuming, so many companies continue to manage the logistics processes utilizing Web-based communication that is much cheaper than other solutions.

We propose a general framework that can be used as a base for a family of e-marketplace models that integrate logistics service providers to companies that need to have their products distributed to costumers. Logistics service providers usually offer one or more services among transportation, warehousing and packaging; we want to deal with providers that offer transportation services. In particular we propose an e-marketplace model capable of step-by-step managing the negotiation of all the participants of the e-market including an optimization analysis for both buyers and sellers of the transportation services.

As we have already pointed out, there are currently two contrasting approaches for managing the operations to connect offers to requests in an e-marketplace: the blackboard approach and a more active one in which the system is capable of finding the best offers-requests matching in order to achieve a certain objective.

In our marketplace, a complex operations research based core system will be capable of managing different possible e-auctions (one-single-shot, two or more round, etc.) where the goods transportation services will be put up for sale by companies and will be offered to distribution organizations. Each delivery service is won by the company that has placed for it the lowest prize bid.

How to place bids in a combinatorial auction is a very difficult task. In freight transportation field carriers are more interested in obtaining contracts for group of lanes instead of sum of individual lanes if they can reduce their empty movement costs by combining these lanes into a tour. Therefore, a distribution company logged to the proposed e-marketplace system will obtain a number of services which help it to evaluate the real transportation costs related to a set of combined lanes and to place bids following an intelligent bidding strategy.

In other words, the proposed e-marketplace guides each participant of the market during the negotiation: evaluating the logistical cost of each choice, helping to place offers, managing the bids, administrating the auctions.

The goal of our marketplace is to find for each auction a combined proposal that tries to minimize the total costs of the buyers and to maximize the total profit margin of the sellers of the services. Therefore, the main task of the marketplace engine is to facilitate transactions, which includes helping to find best matching partners. This means building optimization algorithms into transaction handling processes. Such processes need a set of highly sophisticated algorithms. The methods, needed for the operation of such marketplaces involve substantial amount of computations, that is why a Grid-based implementation is suggested.

As noticed above we want to propose a general framework that may be used as a basis for different type of e-marketplaces models. The structure of this framework is showed in figure 1.

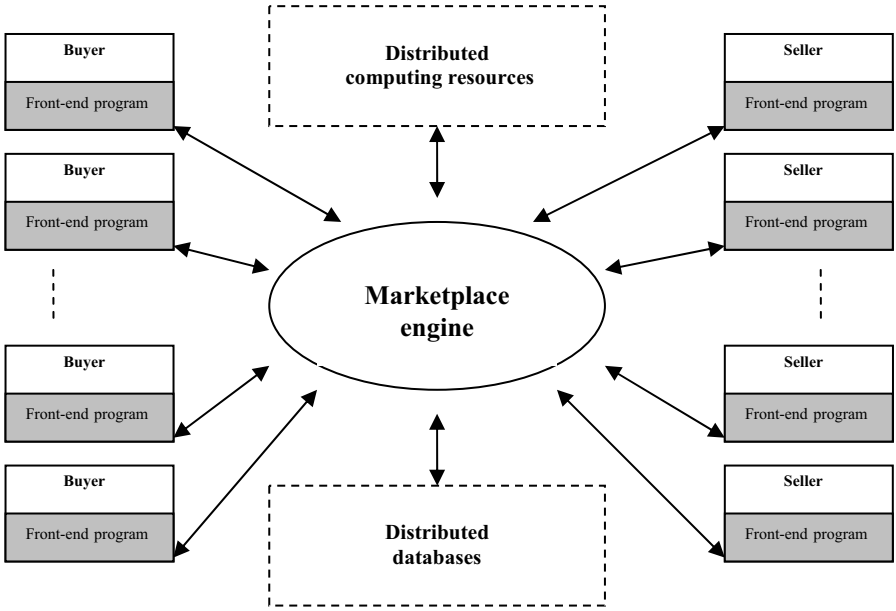


Figure 1. Framework's structure

The marketplace is joint with all the participants of the negotiation by front-end processors.

It is possible to assume that each marketplace's participant acts according to an established business strategy defined by a set of resource and finance planning programs. Each actor of the negotiation communicates with the market environment by its front-end processor, of course this processor operates according with the overall business strategy. The marketplace is joint not only with the front-end processor of all the partners of the arbitration but also computing resources and external databases, distributed all over the grid. The external database containing data related to transportation services; this may regard distances, weather conditions, fuel costs, import-export restriction and so on. The data management is really important to process relevant information in a short time.

According with the definition of framework the marketplace engine and the front-end processors can follow different algorithms according to the different logic of the model. Despite the different possible implementations of this model, some important tasks of the marketplaces are similar, what may change is the way to realize them. First the e-marketplace has to permit the authentication of each participants of the negotiation, then it has to offer software tools to realize the transactions.

A transaction starts with the collection of the service requests: each company needing distribution services (buyer) has to formalize its requests. In particular it has to identify the products, the packages dimensions, the quantities (that is the number of items), the maximum price limits that it wants to spend, the time window in which a delivery has to be completed (that is the earliest and latest completion time of the distribution services) and the origin and destination of each delivery.

All the requests that regard a specific day are collected and put up on sale in an apposite auction. The marketplace engine forwards this information to the transportation services providers (sellers). It also helps each seller to compute the real cost of each distribution service on sale, computing the transportation costs related to all the possible routes related to the transportation requests on the system. In addiction it helps the bidders to define their offers, giving them the possibility to use specific software optimization tools to calculate how to bid in an intelligent manner without having to bid for all the possible group of lanes on sale. Finally, the marketplace engine gives assistance to bidders to select and formalize their bids.

Once all the bids are defined, the WDP problem is solved and the winners of the lanes are computed. Finally, the participant to each e-auction are informed of the results of the transactions.

In the following we aim to detail what is the work to do in the optimization modules inserted in the system architecture of our e-marketplace proposal.

Basically, the system has three optimization modules: Bidding Strategy, Bid Insert and Auction Manager. This means that in these modules NP-Hard optimization problems (their formulation is given below) have to be addressed. The system architecture is given in figure 2.

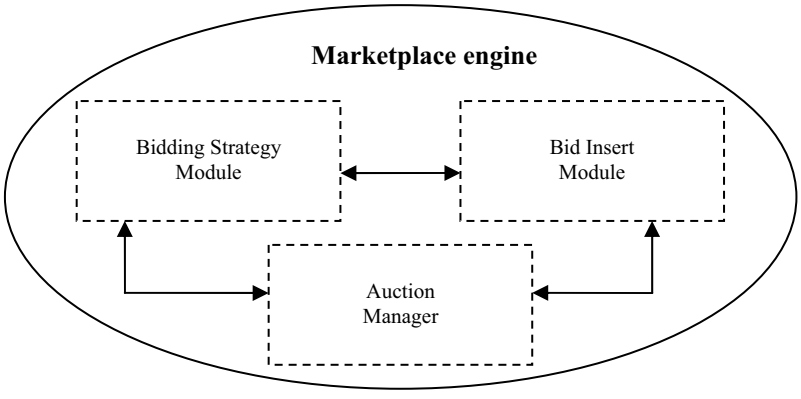


Figure 2. Marketplace engine's architecture

4.1. The Bidding Strategy module

The Bidding Strategy module is a software tool that helps bidders to place their offers. The module addresses a bidding construction problem. This problem has been well formulated by Song and Regan [16] and, substantially, it can be viewed as a particular case of the well known Set Partitioning problem or the Set Covering problem, which are both NP-Hard. The authors propose a valid heuristics for the problem whether in case the bidder has to add new transportation requests in already planned routes and whether it does not have any pre-existing commitments. The power of grid computing may give the possibility to solve those problems to optimality in a wide range of real life instances.

In our idea, the Bidding Strategy module is invoked by distribution companies logged on the e-marketplace and it runs an algorithm that first takes note of all the lanes on sale and asks the carriers to decide if the lanes have to be inserted in already planned routes or not and what are the origins of the vehicles. The algorithm after having computed the route costs related to all possible bids, formulates and solves a Set Covering model, calculating the group of lanes to bid for, in order that all the lanes are inserted in at least one bid and the total empty movement cost is minimized.

The optimization models to address by the Bidding Strategy algorithm are: the bid construction in the absence of pre-existing commitments and the bid construction in the presence of pre-existing commitments.

4.1.1. Bid construction in the absence of pre-existing commitments

In this context, carriers either do not have any pre-committed contracts of current lanes, or they do not intend to integrate new lanes into their current operations. Hence, they are only interested in the combination opportunities among new lanes. This is not unusual in practice as large carriers will run dedicated sub-fleets assigned to individual (large) shippers.

The idea is to make carriers construct bids in such a way that the total operating empty movement cost is minimized. This essentially requires solving a vehicle routing problem. One important approach for solving vehicle routing problem is to formulate vehicle routing problem as a set partitioning problem and to then use a column generation method to obtain exact solutions [49][50].

The first step of this bidding strategy involves using an exhaustive search algorithm to enumerate all routes with respect to routing constraints and treat each route as a decision variable in the set partitioning formulation. For example, a depth first search algorithm can be applied to find all routes satisfying the following constraints:

A route visits each location at most once.

No two empty lanes can occur consecutively in a route (these would be replaced by a single direct empty move).

Additional operational constraints such as maximum route distance or driver work rules.

In this process each new lane is duplicated such that it can be used as an empty lane by other routes. Each constructed route constitutes a candidate bid $y_j \in \omega$: the new lanes in this route form the set of items bid for and its reservation cost determines the bid price and can be calculated based on route length, empty movement cost and carriers' profit margin [24]. The Bid Insert module will be responsible of calculating the

bid price and how to do it will be explained below. The total empty movement cost e_j of each route is calculated and associated with each bid y_j. Calculating this cost is obtained by interfacing the Bidding Strategy module with the internal DB of the distribution company that has called the Bidding Strategy module and considering external data (distances, weather conditions, and so on) stored in the central DB module. All bids are the candidate bids and are inserted as decision variables in the Set Partitioning Problem formulation of the Bid Construction Problem (BCP-SP) as follows:

$$
\begin{aligned}
Min \quad & \sum_{j=1}^{J} e_j y_j & & & \\
s.t. \quad & \sum_{j=1}^{J} b_{ij} y_j & = & u_i & \forall i \in I \\
& y_j & \in & \{0,1\} & \forall j = 1,...,J \\
& b_{ij} & = & \begin{cases} 1 & if \quad lane \quad i \quad is \quad in \quad bid \quad j \\ 0 & otherwise \end{cases} &
\end{aligned}
$$

where y_j is a binary decision variable of candidate bid in set J, i is a new lane in set I, and u_i is the number of loads on that lane. The objective function minimizes the total empty movement cost under an optimal allocation of the new lanes, while the first constraint guarantees that each new lane will be served by exactly one route.

Notice that the number of optimal routes in a solution may exceed a carriers' fleet capacity. However, this problem can be solved by restricting the number of routes selected to be equal to or less than the carriers' fleet size. Also note that, generally, large trucking companies regularly accept contracts for more routes than they can serve and then sub-contract excess demand.

An optimal solution to the BCP-SP problem has three important features.

First, each lane is covered only by one optimal bid so that any two optimal bids are mutually exclusive of new lanes.

Second, combinational bids consisting of collections of new lanes are favoured against single item bids when a complementary relationship exists between these lanes. This implies that a carrier should bid aggressively for these bundles.

Finally, this formulation guarantees that even if only a subset of submitted bids is awarded by the shipper, that subset will still form an optimal solution to this carriers' vehicle routing problem [16].

As pointed out by Song and Regan [16], this formulation could make a company lose good bidding opportunities due to the fact that optimal bids contain mutually exclusive group of lanes. For example, consider that there are three lanes for bid: AB, BA and BCA. Let us assume that the bidder loses BA in auction and the optimal bids in output by the algorithm that solves the BCP-SP problem are{AB; BA} and {BCA}. In this case the bidder has lost also AB. Suppose that the bidder has placed also the bid {AB; BCA} this gives him the possibility to gain AB also if BA is not awarded. Therefore, Song and Regan suggest to use a Set Covering formulation (BCP-SC) of the problem instead of a Set Partitioning one:

$$
\begin{aligned}
Min \quad & \sum_{j=1}^{J} e_j y_j & & \\
s.t. \quad & \sum_{j=1}^{J} b_{ij} y_j \geq u_i & & \forall i \in I \\
& y_j \in \{0,1\} & & \forall j = 1,\ldots,J \\
& b_{ij} = \begin{cases} 1 & if \quad lane \quad i \quad is \quad in \quad bid \quad j \\ 0 & otherwise \end{cases} &&
\end{aligned}
$$

The set covering problem (SC) has been well addressed in literature and many good algorithms have been constructed to reach exact solutions in relatively short time also for real life instances size.

4.1.2. Bid construction in the presence of pre-existing commitments

It is common for carriers to have contracts serving multiple customers. The addition of new lanes may be normally integrated into a carriers' current operations. Therefore, carriers have to consider how to insert these new lanes into their current routes. It is certainly true that new lanes represent new opportunities for carriers but, on the other hand, pre-existing plans might need to be protected.

These new opportunities can be inserted into the BCP-SC formulation [16]. The model is the following:

$$
\begin{aligned}
Min \quad & \sum_{j=1}^{J_1} e_j y_j & & \\
s.t. \quad & \sum_{j=1}^{J_1} b_{ij} y_j \geq u_i & & \forall i \in I \\
& y_j \in \{0,1\} & & \forall j = 1,\ldots,J_1 \\
& b_{ij} = \begin{cases} 1 & if \quad new \quad lane \quad i \quad is \quad in \quad bid \quad j \\ 0 & otherwise \end{cases} && \\
& b_{kj} = \begin{cases} 1 & if \quad current \quad lane \quad k \quad is \quad in \quad bid \quad j \\ 0 & otherwise \end{cases} &&
\end{aligned}
$$

Now the set of candidate bids is J_1 and we have $J \subset J_1$ in addition to those bids in J consisting solely of new lanes and/or empty lanes, J_1 also includes those bids with a combination of current lanes and/or new lanes and/or empty lanes. I is the set of new lanes and K is the set of pre-existing lanes. The objective is still to minimize the total empty movement costs. The first set of constraints guarantees that each new lane is covered by at least one bid and the second set guarantees the inclusion of current lanes in pre-existing routes.

In an optimal solution, if a selected bid only includes current lanes and/or empty lanes, this is a routing plan for pre-existing commitments; if it also includes new lanes, these new lanes will constitute an atomic bid.

4.2. The Bid Insert Module

Several researchers notice that it is computationally difficult for the bidders to valuate the values or costs associated to each possible item to bid for [15][16]. Contrary to the assumption in classic auction theory that bidders know these information a priori, bidders in a combinatorial auction have to allocate resources to determine the values to associate to their bids. In our proposal a specific module is devoted to calculate costs and bid prizes of group of lanes.

This module is called Bid Insert and its aim is to help bidders in developing their bids for the auction. For each item that a carrier wants to bid for (which may be the items given in output by the previously described Bidding Strategy module or directly decided by the bidder) the Bid Insert module formulates and solves a Travelling Salesman Problem (TSP). TSP is a particular logistics problem where the goal is to find the shortest route to visit a collection of cities and return to the starting point. The problem is formulated on a graph $G\ (V,A)$ where V is the set of nodes (cities) and A is the set of arcs, i.e. the set of connections between nodes. To each arc $(i,j) \in A$ is associated a cost c_{ij} . The costs of the arcs are input data that the module can know interfacing itself with the databases distributed all over the Grid environment.

The optimization model for TSP is given in the following:

$$
\begin{aligned}
Min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} & & & \\
s.t. \quad & \sum_{i\in V\setminus\{j\}} x_{ij} & = & 1 & \forall j \in V \\
& \sum_{j\in V\setminus\{i\}} x_{ij} & = & 1 & \forall i \in V \\
& \sum_{i\in S}\sum_{j\notin S} x_{ij} & \geq & 1 & \forall S \subset V, |S| \geq 2, \forall (i,j) \in A \\
& x_{ij} & \in & \{0,1\} & \forall (i,j) \in A
\end{aligned}
$$

The problem is NP-Hard and it needs massive calculations resources to be solved in acceptable time for real-world instances.

Once a TSP solution is found, the module knows the cost of the item given in input and it is able to calculate the bid price simply interrogating the databases distributed on the Grid. The bid price can be calculated basing on route length, empty movement cost and carriers' profit margin. Finally, the Bid Insert module returns the calculated bid price to the bidder along with the cost of the group of lanes received in input. The bidder is now able to choose if accepting the Bid Insert suggested bid price or insert a bid with a different associated value.

4.3. The Auction Manager Module

The Auction Manager is the module responsible for the determination of the winner bids in an auction planned on the system. The optimization problem to address by the Auction Manager is the WDP. First, all the lanes are inserted and put up on sale, then all the bids are stored and a particular Linear Programming optimization problem is formulated and addressed. In the case of freight transportation the model is given in the following. It is a *reverse combinatorial auction*, where buyers (shippers) require a set of G items (freight transportation requests between a specific origin and a particular destination) to many possible sellers (carriers). A bid b is defined as $b=(S_b,p_b)$, where S_b is a subset of items and p_b represents the prize to pay by shippers to a seller to have the products distributed. Let B be the set containing all the bids and x_b $\forall b \in B$ a decision variable that is equal to 1 if b is a winner bid, 0 otherwise. The goal is to find the bids that minimize the total prize to pay by shippers.

The optimization model is:

$$
\begin{array}{lrcl}
Min & \sum_{b \in B} p_b x_b & & \\
s.t. & \sum_{b \in B} \delta_{i,S_b} x_b & = & 1 \quad \forall i \in G \\
& x_b & \in & \{0,1\} \quad \forall b \in B
\end{array}
$$

where $\delta_{j,S_b} = 1$ if $i \in S_b$, 0 otherwise.

As it easy to see from the formulation the WDP to solve belongs to the NP-Hard class of problems and therefore grid computing techniques are very helpful to obtain solutions for real life instances.

5. Conclusions

The integration between electronic marketplaces and supply chain management is currently a hot topic. One instance of merging these two concepts are e-marketplaces for logistics services. They act as electronic agents and they aim to connect logistics service providers to companies that need to have their products distributed to costumers. Over the last few years, the electronic logistics marketplaces have emerged as new business models with the potential of having enormous influence over the way that transactions are carried out, relationships are formed, supply chains are structured and profit flow are operated. Enabled by the web technology, they may provide advantages in low cost inter-organization information connectivity, real time visibility and flexible partnership configurations between buyers and sellers.

Buyers and sellers could leverage electronic logistics marketplaces by collaborating on a single platform and eliminating the complex and costly integration of different inter-organizational systems; they could also use electronic markets to reach wide resources, or to collaborate with other similar company. Summarizing,

virtual logistic marketplaces encompass a wide range of tactics designed primarily to drive sales and grow business of all their participants.

Over the years, we have observed a proliferation of e-marketplaces for business services. Logistics services are not excluded from this process. However, we think the potential impact of electronic logistics marketplaces is more significant than simply transacting the buying and selling of logistics services online. Actually, this is what the logistics e-marketplaces offer currently.

The benefits of incorporating logistics services into either horizontal or vertical marketplaces are only starting to be experimented. In particular we think the integration between logistics e-marketplaces and the grid technology could be very important for this exploration.

The main advantage of this integration could be the possibility to realize an optimized logistics e-marketplace. It requires large computational power (a great number of efficient logistics optimisation algorithms ask for adequate computing resources); distributed computing resources may be used to give quick and efficient optimized solutions to all the participants of the virtual market. In particular, grid technology offers the possibility to distribute the amount of calculations among a set of distributed resources; to realize an efficient and effective management of distributed data, and to integrate existing value-added services.

The aim of this research has been to design a general framework that can be used as a basis for a family of e-marketplace models that put together logistics service providers to companies that need to have their products shipped to their costumers.

This research has focused on the suitability of grid technology for the development and the use of an optimized logistics e-marketplaces. Evidence from this study indicates that grids are appropriate for the designed e-marketplace model and their power should be actively used in order to exploit the great potentiality of these powerful systems.

Future research will be conducted following two different directions: the first one regards a detailed analysis of the mathematical models and the development of distributed algorithms for the optimization problems on which the proposed marketplace engine is based, the other regards the implementation of the designed e-marketplace architecture.

References

[1] Ghiani, G. Laporte, G., Musmanno, R., Introduction to Logistics Systems Planning and Control, Wiley, 2004.

[2] Lim, B., Wen, J., Web Services: An Analysis of the Technology, its Benefits, and Implementation Difficulties, *Information Systems Management* **20 (2)** (2003), 49-57.

[3] Foster I., Kesselman, C., The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 1999.

[4] Iamnitchi, A., Foster, I., *A problem-specific fault-tolerance mechanism for asynchronous, distributed systems*, Proceedings of the ICPP'00, International Conference on Parallel Processing, Canada (2000), 4–14.

[5] Goux, J.P., Kulkarni, S., Linderoth, J.T., Yoder, M.E., *Master–worker: an enabling framework for applications on the computational grid*, Cluster Computing **4** (2001), 63–70.

[6] Anstreicher, K., Brixius, N., Goux, J.P., Linderoth, J.T., *Solving large quadratic assignment problems on computational grids,*Mathematical Programming, Series **B 91** (2002), 563–588.

[7] Aida, K., Natsume, W., Futakata, Y., *Distributed computing with hierarchical master–worker paradigm for parallel branch-and-bound algorithm*, Proceedings of the CCGrid'03, International Symposium on Cluster Computing and Grid, Japan, (2003), 156–162.

[8] Drummond, L.M.A., Uchoa, E., Gonçalves, A.D., Silva, J.M.N., Santos, M.C.P., Castro, M.C.S., A grid-enabled distributed branch-and-bound algorithm with application on the Steiner Problem in graphs. *Parallel Computing* **32(9)** (2006), 629-642.
[9] Bendjoudi, A., Melab, N., Talbi. E.G, *A Parallel P2P Branch-and-Bound Algorithm for Computational Grids*, Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07), 749-754.
[10] Staab S., Van der Aalst W., Benjamins V. R., Sheth A., Miller John A., Bussler C., Maedche A., Fensel D., Gannon D., Web Services: Been There, Done That?, *IEEE Intelligent Systems* (2003), 72 - 85.
[11] Foster, I., Keselman C., Nick, J., Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems integration, 2002. http://www.globus.org/research/papers/anatomy.pdf
[12] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., Maquire, T., Sandholm, T., Snelling, D., Vanderbilt, P., Open Grid Services Infrastructure (OGSI) Version 1.0, June 2003, http://www.globus.org/research/papers/Final_OGSI_Specification_V1.0.pdf.
[13] http://www.inttra.com.
[14] http://www.telerouteitalia.com.
[15] Parkes, D.C., Optimal auction design for agents with hard valuation problems. In: Moukas, A., Sierra, C., Ygge, F. (Eds.), Agent Mediated Electronic Commerce II: Towards Next-Generation Agent-Based Electronic Commerce Systems (LNAI 1788). Springer, Berlin, 2000.
[16] Song, J., Regan, A. C., Approximation algorithms for the bid construction problem in combinatorial auctions for the procurement of freight transportation contracts. *Transportation Research Part B* **39** (2005), 914-933.
[17] De Vries, S., Vohra, R.V., Combinatorial auctions: a survey, *INFORMS Journal on Computing* **15 (3)** (2003), 284–309.
[18] Caplice, C., An Optimization Based Bidding Process: A New Framework for Shipper-Carrier Relationship. Ph.D. Thesis, School of Engineering, MIT, 1996.
[19] Ledyard, J., Olson, M., Porter, D., Swanson, J., Torma, D., The first use of a combined value auction for transportation services, *Interfaces* **32 (5)** (2002), 4–12.
[20] Elmaghraby, W., Keskinocak, P., Technology for transportation bidding at The Home Depot, Teaching case, School of Industrialand Systems Engineering, Georgia Institute of Technology, 2000.
[21] Elmaghraby, W., Keskinocak., P., Combinatorial auctions in procurement. Technical report, The Logistics Institute, Georgia Institute of Technology, Atlanta, GA, 2002.
[22] Caplice, C., Sheffi, Y., Theory and practice of optimization based bidding for motor carriers transport services, *Journal of Business Logistics* **24 (2)** (2003), 109–128.
[23] Sheffi, J., Combinatorial auctions in the procurement of transportation services, *Interfaces* **34 (4)** (2004), 245-252.
[24] Song, J., Regan, A. C., Combinatorial auctions for transportation service procurement: the carrier perspective, *Transportation Research Record* **1833** (2004), 40–46.
[25] Song, J., Nandiraju, S., Regan, A.C., Optimization Models for Auctions for Transportation Service Contract Procurement, UCI-ITS-WP 0425., 2004.
[26] Nisan, N., Bidding and Allocation in Combinatorial Auctions, ACM conference on Electronic Commerce, 2000.
[27] Sandholm, T., Algorithm for optimal winner determination in combinatorial auctions, *Artificial Intelligence* **135** (2002), 1–54.
[28] Abrache, J., Bourbeau, B., Crainic, T.G., Gendreau, M., A New Bidding Framework for Combinatorial E-Auctions, *Computers and Operations Research* **31 (8)** (2004), 1177 – 1203.
[29] Rothkopf, M., Pekec, A., Harstad, R.M., Computationally manageable combinational auctions. *Management Science* **44 (8)** (1998), 1131–1147.
[30] Day, R.W., Raghavan, S.. CAMBO: Combinatorial Auctions using Bids with Order, working paper, University of Maryland, 2003. http://www.math.umd.edu/~rwday/cambo.pdf
[31] Cramton, P., Spectrum auctions, in Cave, M., Majumdar, S., Vogelsang, I. (Eds.), *Handbook of Telecommunications Economics* (2002), Elsevier Science B.V., Amsterdam, 605–639.
[32] Banks, J.S., Ledyard, J.O., Porter, D.P., Allocating uncertain and unresponsive resources: an experimental approach, *Rand Journal of Economics* **20** (1989), 1–22.
[33] Bykowsky, M.M., Cull, R.J., Ledyard, J.O., Mutually destructive bidding: the FCC design problem, *Journal of Regulatory Economics* **3(3)** (2000), 205–228.
[34] DeMartini, C., Kwasnica, A.M., Ledyard, J.O., Porter, D., A New and Improved Design for Multi-Object Iterative Auctions, Social Science Working Paper No. 1054, California Institute of Technology, 1999.
[35] Parkes, D.C., Ungar, L.H., Iterative combinatorial auctions: theory and practice, in *Proceedings of 17th National Conference on Artificial Intelligence (AAAI-00)* (2000), 74–81.

[36] Parkes, D.C., Ungar, L.H., Preventing strategic manipulation in iterative auctions: proxy agents and price adjustment, in *Proceedings of 17th National Conference on Artificial Intelligence (AAAI-00)* (2000), 82–89.
[37] McAfee, R.P., McMillan, J., Auctions and bidding, *Journal of Economic Literature* **25 (2)** (1987), 699–738.
[38] Larson, K., Sandholm, T., Computationally Limited Agents in Auctions, International Conference on Autonomous Agents, Workshop on Agent-based Approaches to B2B, Montreal, Canada, (2001), 27–34.
[39] Conen, W., Sandholm, T., Preference Elicitation in Combinatorial Auctions: Extended Abstract, ACM Conference on Electronic Commerce (ACM-EC), Tampa, FL (2001).
[40] Jones, J., Koehler, G.J., Combinatorial auctions using rule-based bids, *Decision Support Systems* **34 (1)** (2002), 59–74.
[41] Parkes, D.C., Ungar, L.H., Foster, D.P., Accounting for cognitive costs in on-line auction design, in Noreiga, P., Sierra, C. (Eds.), Agent Mediated Electronic Commerce (LNAI 1571). Springer-Verlag, (1999), 25–40.
[42] Fisher, M., Vehicle routing, in Ball, Magnanti, Monma, Nemhauser (eds.), *Handbooks in Operations Research and Management Science* **8**, Network Routing, (1995), 1–34.
[43] Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F., Time constrained vehicle routing, in Ball, Magnanti, Monma, Nemhauser (Eds.), *Handbooks in Operations Research and Management Science* **8**, Network Routing, (1995), 35–140.
[44] Powell, W.B., Jaillet, P., Odoni, A., Stochastic and dynamic networks and routing. In: Ball, Magnanti, Monma, Nemhauser (Eds.), *Handbooks in Operations Research and Management Science* **8**, Network Routing (1995), 141–296.
[45] Delaitre, T., Goyeneche, A., Kiss, T., Winter, S.C., Publishing and Executing Parallel Legacy Code using an OGSI Grid Service, *Lecture Notes in Computer Science* **3044** (2004), 30-36.
[46] Delaitre, T., Goyeneche, A., Kacsuk, P., Kiss, T., Terstyanszky, G.Z., Winter, S.C., GEMLCA: grid execution management for legacy code architecture design, in *Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04)* (2004), 477-480.
[47] Delaitre, T., Kiss, T., Goyeneche, A., Terstyanszky, G., Winter, S., Kacsuk, P., GEMLCA: Running Legacy Code Applications as Grid Services, *Journal of Grid Computing* **3** (2005), 75-90.
[48] Tibben-Lembke Ronald S., Rogers Dale S., Differences between Forward and Reverse Logistics in a Retail Environment, *Supply Chain Management: An International Journal* **7 (5)** (2002), 271 - 282.
[49] Desrochers, M., Desrosiers, J., Solomon, M., A new optimization algorithm for the vehicle routing problem with time windows, *Operations Research* **40 (2)** (1992), 342-354.
[50] Bramel, J., Simchi-Levi, D., On the effectiveness of set covering formulations for the vehicle routing problem with time windows, *Operations Research* **45** (1997), 295-301.

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Challenges Facing Production Grids

Ruth PORDES
Fermi National Accelerator Laboratory[1],
PO Box 500, Batavia, Illinois, USA

Abstract: Today's global communities of users expect quality of service from distributed Grid systems equivalent to that their local data centers. This must be coupled to ubiquitous access to the ensemble of processing and storage resources across multiple Grid infrastructures. We are still facing significant challenges in meeting these expectations, especially in the underlying security, a sustainable and successful economic model, and smoothing the boundaries between administrative and technical domains. Using the Open Science Grid as an example, I examine the status and challenges of Grids operating in production today.

Keywords: Global Communities, Grids, Distributed Computing, High Throughput Computing, Opportunistic Computing

Introduction

Modern Grid infrastructures transfer up to a terabyte of data and run tens of thousands of processing jobs a day supporting not only "early adopter" physical science applications but also a broad range of research including nanotechnology, drugs and disease research, and the social sciences.

There are several large national and international Grids, for example, Enabling Grids for EsciencE (EGEE)[1], GridPP[2], Nordic DataGrid Facility[3], Open Science Grid[4], and TeraGrid[5]. These Grids provide access to and sharing of as much as five hundred Teraflops of processing power. They include as many as two hundred independent physical sites. They provide data storage using tape and disk caches of as much as ten PetaBytes. They support data movement across a range of production and research networks from a hundred Mbit to several tens of GBits. Each Grid serves from ten to a hundred independent user communities each consisting from a single to a hundred users, and many of which act across multiple infrastructures.

In 2008, the largest user communities are expected to run more than twenty thousand jobs a day across local and remote resources. This will involve moving data at sustained rates of up to ten Gigabit/second with latencies for the transfer of Terabyte datasets of less than a day. Each community will support hundreds of users doing data analysis in

[1]Fermilab is supported by the U.S. Department of Energy under contract No. DE-AC0276CH03000.

simultaneously. As these distributed facilities are relied on more and more, there is an accompanying expectation that they will be as robust, capable and available as local data centers.

Modern researchers depend on a rapidly increasing amount of computation and electronic data to make progress. Individuals, small research teams, and university groups, need to gain access to and learn to use remote resources. Usable common distributed infrastructures lower the barriers for these groups to benefit from distributed facilities as well as increase local productivity by enabling sharing of resources across the campus or local region.

1. The Context

The challenges facing production Grids today are threefold: First to provide high reliability, high throughput, scalable, multi-user, distributed data centers which operate around the clock and around the world; second to provide the security, technologies and infrastructure to serve an increasingly large and demanding community of researchers, educators, commercial companies and the general public; and third provide usable services that facilitate the entry of new participants in the use of distributed computational infrastructures.

Present experience where the Grid is driven by and embedded with the end user communities is encouraging in the success of matching expectation to realization and in the effectiveness of the operating infrastructure. This inclusion of all the actors – the users, the facility owners, and the technology providers, is a key component of the success. One of the larger distributed facilities, the Open Science Grid (OSG) Consortium, provides such an example. The scientists from the internal stakeholder communities – especially the Large Hadron Collider (LHC)[6] ATLAS and CMS experiments and the Laser Interferometer Gravitational Wave Observatory (LIGO)[7] – are immersed into the project in all aspects of the management and technical program. Similarly the computer science researchers and information technologists, who provide the software and services on which the infrastructure relies, participate fully in the leadership and all activities of the Consortium.

1.1. The Open Science Grid Distributed Facility

The Open Science Grid distributed facility, shown in Figure 1, includes: sites which provide shared storage and processing resources; communities, or Virtual Organizations[8], which include the users of the facility working in collaborating groups; and the common infrastructure which provides services to integrate and operate the facility as a whole. The infrastructure also provides gateways to other Grids.

The following principles[9] guide the Open Science Grid's implementation:

The owners of a site are responsible for, control, and manage access to, their resources.

The community is responsible for its users and applications and their access to and use of the resources.

- The OSG staff is responsible for the common services and for federating with other distributed facilities including campus, regional, and international infrastructures.

Such a facility can only thrive in an environment and with a value proposition where:

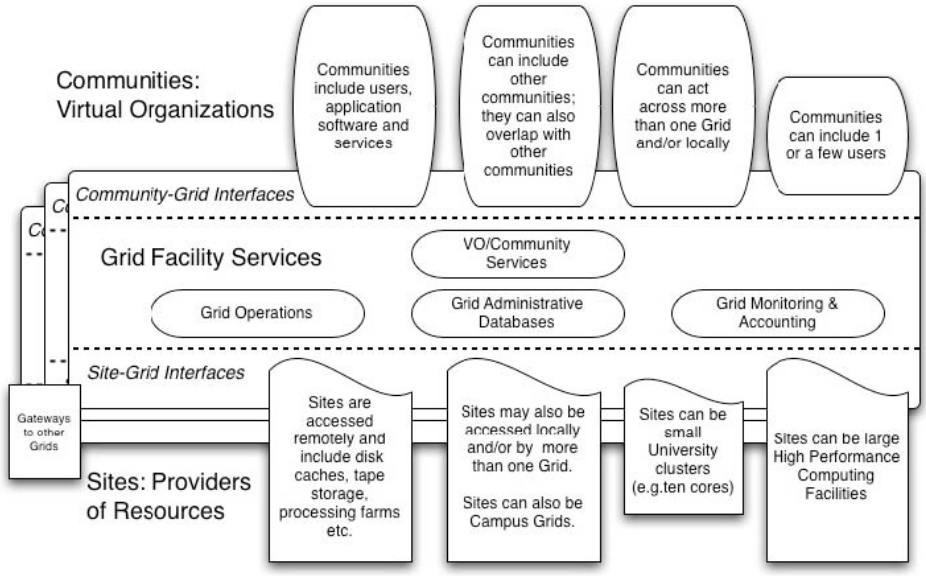


Figure 1: The OSG Facility

The economic model provides effective sharing so that the benefit experienced from the ensemble of resources is sufficiently greater than that delivered by the sum of each individual site.

The security, policy and trust infrastructures demonstrably protect all participants from unintended as well as deliberate misuse and attack.

OSG today is a production infrastructure providing access to more than fifty processing sites and ten grid accessible storage sites, where space can be managed remotely, used by more than twenty VOs. The processing sites provide access to about thirty thousand cores providing over a hundred teraflops peak usage, with the size of the sites ranging from a few tens to a few thousand cores each. While the majority of the processing and storage resources are provided for the owners of the resource, on average more than ten percent are provided on a regular basis for other users – and on many days the "opportunistic" usage is significantly greater. In parallel with the production system is an integration or test infrastructure of about fifteen processing and two storage sites. The processing sites internally have a significant amount of both shared and local storage, with the total amount of disk accessible storage being of the order of a few petabytes.

1.2. Implementation Architecture

A production grid enables multiple VOs to use and share resources, software and services. The definition of an OSG VO includes not only the people but also the services and the resources used by the organization. And a VO may include sub-VOs within it. In the OSG context each VO acts as a "community grid" overlaid over one of more grids – there being a well-defined scope for the services, policies and procedures which make up the organization.

Within a grid there are a set of well-defined common interfaces between heterogeneous implementations for each component. These interfaces allow the distributed data center to act as a system and provide a manageable and smooth process for adding resources and new applications. Our architecture, which is typical for production grids today enables a resource to be accessed and used through multiple interfaces—locally through existing "non-grid' means and/or through other grids such as local shared campus infrastructures etc. Similarly, the implementation architecture is cognizant that any VO may be using multiple infrastructures simultaneously and may have a deep set of (sometimes complex) shared software and services that are specific to the VO and operate across these infrastructures. Other architectural aspects include:

Shared resources provide not only agreed levels of use but also provide "opportunistic" access to otherwise unused resources.

Each service or resource manages their own interfaces.

The production facility supports multiple versions of the infrastructure simultaneously.

There are no single points of failure.

The distributed facility supports incremental upgrade and extensions.

All components adhere to the facility security requirements and can trace their use and access to a responsible person. Access may be denied (to a VO or user) based on security as well as contract and policy requirements.

Latency as well as performance needs influence the implementation.

1.3. Lacks in Implementation

Current implementations of the architecture are lacking in several dimensions. Extending the current practices can solve some of these, but others need research into new methods and techniques:

The ability of the services and components to defend themselves against overload and unintentional misuse by the application software.

The impossibility of testing all configurations in such a heterogeneous environment before putting new software into production.

The lack of software to support dynamic sharing of storage on a distributed set of resources.

The lack of the infrastructure to support recursive sub-VOs in a manner equivalent to VOs, including the OSG VO itself.

End-to-end communication of, response to, and diagnosis of downtimes, failures and errors.

2. The Value Proposition

Large scientific collaborations control and own their vertically integrated data handling and analysis systems. This is essential to give them the ability to manage and make trade offs in the performance, functionality and scale of their systems to meet their scientific priorities and goals. And the nature of scientific collaboration today is intrinsically collaborative and widely distributed. Reliance on common infrastructures such as the OSG only succeed when there is a clear value which results in a reduction of effort or an increase in total throughput without compromising the end goals.

An example benefit that has been widely acknowledged by the communities is the integration, testing and monitoring of the suite of software and the deployed system. Another is the sharing of expertise and gradual acceptance of common needs and software that can be shared. This latter is a cultural change, which can only come gradually. It is an example of one of the longer-term values of common and collaborative grids that is difficult to quantify but should be included in the tally.

Below we list some the value and benefits through deploying production grids for the science and research community:

Distribution of peak demands across a larger ensemble of resources and on average increase the % usage of available compute and storage cycles;

Reduction in the effort needed to adopt and use distributed systems;

Reuse of software and reduction of effort applied to duplication;

On the ground communication of knowledge and experience;

Reduction in risk of failure due through a broader range of available experts and systems.

Using OSG as the example, the benefits indeed come from reducing the risk in and sharing support for large complex systems, which must be run for many years with a workforce whose availability is significantly shorter than the lifetime of the application projects. Leverage of the expertise and support for such systems to enable new communities to more easily participate in distributed science is an important additional benefit. This includes:

Savings in effort for integration, system and software support,

Opportunity and flexibility to distribute load and address peak needs.

Maintenance of an experienced workforce in a common system.

Lowering the cost of entry to new contributors.

Enabling of new computational opportunities to communities that would not otherwise have access to such resources

Clearly, the existence of a shared virtual facility, aka grid, does not obviate the need for the purchase of hardware and building of computational facilities themselves. In fact when one calculates the basic costs of purchasing and maintaining hardware locally and compare it to the cost to interface and share say 10% of remote resources that one does not own, the results do not favour the latter. A more relevant calculation is the total lifetime cost of building and maintaining complex systems that scientists and researchers can effectively use.

2.1. Benefits from a Common, Integrated Software Stack

Most production grids provide a reference software stack for use by resource owners and application communities to not only ease their participation with the infrastructure but also to ensure a robust, reliable foundation to which the resources interface and on which the applications can act. This testing, large-scale use and support for this software that is demonstrably “used under fire” is one of the values of a recognizable common infrastructure, which is widely deployed.

Most scientific and research grids today rely on the security and execution management technologies from Condor[11] and Globus[12]. In addition, several rely on the additional extensions and integration offered by the OSG software releases – the Virtual Data Toolkit

(VDT) [10]. The VDT includes about thirty additional modules, including components from other computer science groups, the Enabling Grids for EScience (EGEE), DOE Laboratory facilities (Brookhaven, Fermilab and LBNL), the application communities themselves, as well as standard open source software components such as Apache and Tomcat. And, while TeraGrid does not rely on VDT per se, OSG and TeraGrid align their versions of the Condor and Globus software to ensure ongoing interoperability. The EGEE gLite and TeraGrid CTSS software stacks are similar to the VDT in goals and scope, and provide an integrated set for those grids that interoperate with the VDT as well as other software systems.

Once installed on a site the VDT software enables support for remote job submission and local execution, access to remote storage and sharing of local storage, and data transfer into and out of the site. The site also has services to help the administrators manage priorities and access between VOs, support policies within the VOs, and can participate in the OSG monitoring, validation and accounting services.

The VDT also provides client libraries and tools for the applications to use to access the distributed resources and services.

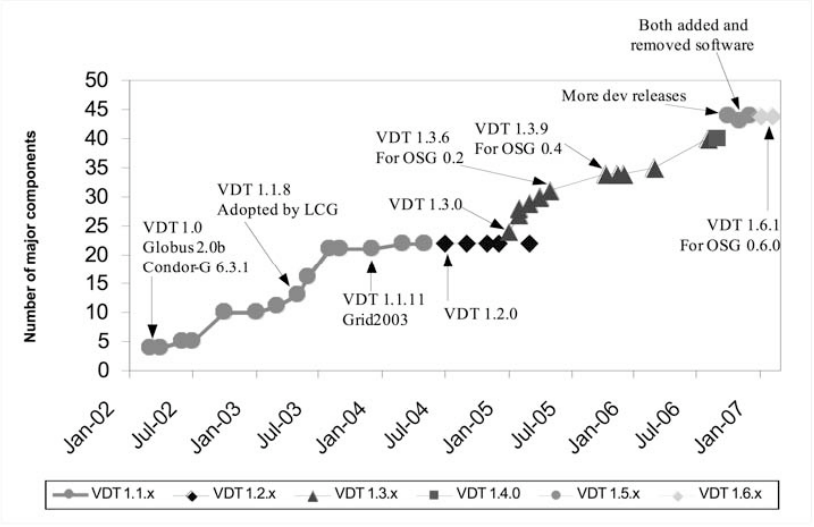


Figure 2: Timeline of VDT Releases

It is essential to maintain the operation and robustness of the production grid infrastructure to changes in software versions and capabilities. Production grids invariably institute a careful process in deploying new software and OSG is a prime example. Before a new release of the VDT, all components are built up to ten Linux platforms (as well as some limited builds on AIX and MacOSX) using the NSF Middleware Initiative build and test facility [13] and, by reusing builds, the VDT tested on up to fifteen platforms. The validity of the integrated release is checked carefully. For example many components use Apache web server and we check that usage is compatible. The integrated release is validated on a small number of well-controlled sites to check it works in a distributed environment. It is

then in at scale on a parallel grid to the production facility. In the OSG case this is a >15 site infrastructure, the Integration Testbed (ITB). The system tests only the individual services and components but also that the system and VO applications work reliably for long periods.

Most, if not all, the research communities' domain specific software and services layered above the production grid software stack. The goal is for them to install their software dynamically using Grid methods and either compile and leave the executables on each site, or move the executable for each job submission. Integration, or pre-production, grid goals are invariably to test the complete matrix of horizontally integrated grid software and vertically integrated VO software.

Software infrastructures provided by production grids also provide tools for incremental installation of new releases and patches, verification of installed configurations, and for functional testing of the sites. Security and robustness of the software as deployed is of paramount concern and an ongoing area of activity on alls production infrastructure. While progress is being made the following areas still merit research and development to make the support and use of these grids more secure, usable and dependable:

- Tools to audit the use of and probe the software interfaces to determine overload and "denial of service" conditions.

2.1.1. Challenges with Software Evolution and Support

The following needs for software support have been shown to be ongoing challenges:

Quick and efficient patches of the software and redeployment in response to security notifications.

Balancing the amount and effort spent on testing with the need to get new services "to the user" quickly. The stability of the resulting infrastructure becomes at risk since testing invariably does not cover the full set of usage patterns.

Adding functionality (driven by the user communities) balanced with the need to make the software stack minimal and low impact.

Integration of diverse software components from multiple software suppliers with different levels of development maturity and different release cycles.

3. An Economic Model

Computing and storage resources owners make their resources accessible to the OSG shared distributed facility and retain control of the management, use and policies of these resources (except of course in the response to security incidents) The OSG Consortium makes no policy on their use except that each resource contributor is expected to support at least one OSG application which is not the one owning the resources --otherwise this would obviate the need or interest in being part of the OSG -as well as supporting the OSG administrative VOs for monitoring, accounting, validation.

Requests for production running are brought to the Executive Board to assess the technical needs and to the Council for an understanding of resource availability and policy. The Consortium members define policies that allow users of the OSG to take advantage of available and otherwise unused cycles. The economic model of OSG has the goal that the ensemble offers an overall higher peak and average throughput than would come from the sum of individual and separate use of each of the resources; that sharing of resources is a

win-win situation; and that we encourage sites to define policies for opportunistic use by a broad set of OSG member VOs wherever possible. In the long run OSG can only work if people give as well as receive. At the moment there are sufficient resources and the sum of the VO needs and ability to run on sites is such that OSG Council members have not yet had to take the hard decisions that we know are coming when there is oversubscription and there are many people knocking on the door.

3.1. A Practical example: Opportunistic Use

D0 is one of the two currently running experiments at the Tevatron in Batavia, Illinois. The experiment currently has more than two and a half petabytes of raw and processed data on archival tape. D0's own resources are committed to the processing of newly acquired data and analysis of these datasets as they are processed. In November 2006 D0 asked to use fifteen to twenty thousand CPUs for up to four months for re-processing of an existing dataset (~500 million events) to deliver science results for the summer conferences in July 2007. The OSG Executive Board estimated there were currently sufficient opportunistically available resources on OSG to meet the request; we also reviewed that the local storage and I/O needs could be met. The Council members agreed to contribute resources to meet this request. D0 had several months of smooth production running using more than a thousand CPUs and met their goal by the end of May.

The steps to achieving this included: D0 testing of the integrated software system from December to February; OSG staff and D0 working closely together as a team to reach the needed throughput goals during February and March; working through detailed problems at more than twelve OSG sites as well as sites on the EGEE, Canada and UK grids; support for sustaining the throughput during April and May. The goal was achieved by the end of May. OSG contributed over half of the total events processed (286 million events) using more than two million CPUhours (see Figure 3) of opportunistically available cycles by three hundred thousand jobs, the average job length being between six and eight hours. During the reprocessing 48TeraBytes of data were moved from the central archive at Fermilab to the OSG sites and 22TeraBytes was moved from the OSG reprocessing sites back to the central archive.

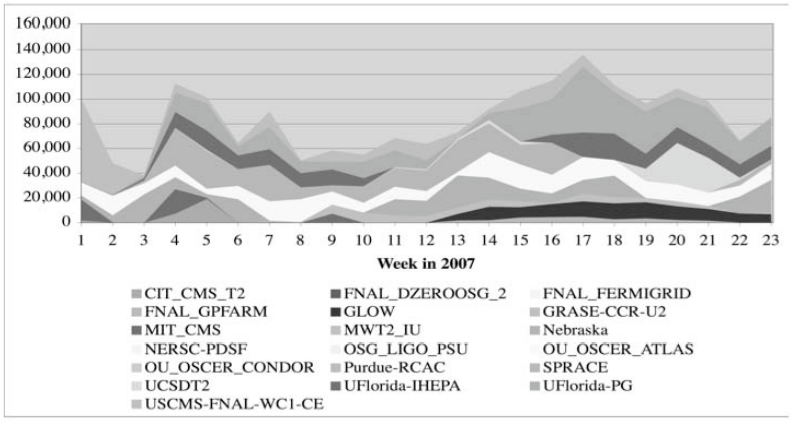


Figure 3: D0 CPUHours/Week on OSG in 2007

3.1.1. Lessons Learned

This production activity was a very instructive experience in learning how OSG as an organization, as well as an operating infrastructure, can handle such requests. The ramp up time, as well as the unique problems experienced at each site as the VO's application was commissioned, can clearly benefit from improvement. Additional problems and overheads were experienced scaling the whole system up to the throughput needed. As explained above we faced two main problems: Overall root cause analysis and troubleshooting of the quite different hardware configurations at the sites—especially data storage architectures—was a challenge; and during the steady state operations phase we found inefficiencies in our processes for notification and responding to site downtimes and problems.

While the activity was a success there are several issues that we are taking under consideration for the future:

Sites do provide substantial sharing of their resources and VOs can rely on significant effective throughput from resources they do not own.

Ongoing teamwork between the application and infrastructure groups is essential in such a complex software and hardware environment.

We need to start preparing now for the future where the resources on OSG are oversubscribed and prioritization across the infrastructure is needed.

4. Production Grid Services

4.1. Security

The security challenges faced by distributed data centers extend beyond those for a fire walled single site. Most grids define the site and VO management as responsible for the security of the resources and services they own. The VO management is responsible for ensuring their individual users follow best practice and are aware of security and its needs.

The grid organization is responsible for the risk analysis, assessment and auditing of the assets it owns – the grid-wide services and the software it provides. Also the grid organization monitors and acts on risk and identified incidents across the whole infrastructure. In general, production grids do have active security activities that include security management, operational and technical components, agreed upon scope and authorities [14]. Communication of and timely response to security alerts is essential and the possibility of a major incident causes most grid managers loss of sleep.

As an indication of scale, OSG has responded to more than fifteen security alerts that involve software vulnerabilities over the past year. These invariably result in new software releases and significant effort is spent developing and distributing these patches in a timely fashion. Communication across the software development teams early in the process is crucial for fast in depth analysis of the problem, validation of proposed solutions and non-perturbative deployment.

To identify its users, resources and services the largest production grids in action today, (including EGEE and OSG) support X509 user, host and service certificates allocated through one of the International Grid Trust Federation (IGTF) [15] accredited Certificate Authorities. The grids provide services that extend these certificates by adding information about the VO the use of the certificate will be associated with for a particular set of transactions, and in some cases a finer granularity (or role) within the VO which will be used to determine the access policies and priorities that will be applied to the transactions

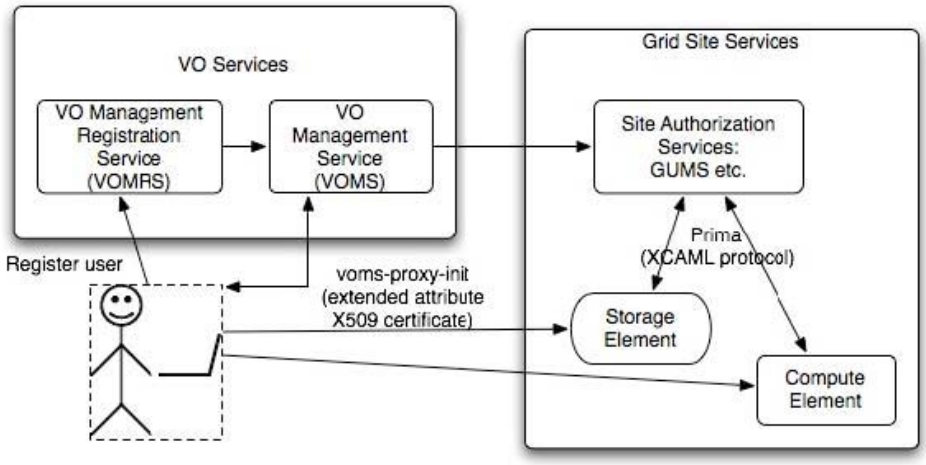


Figure 4: The OSG Authorization Infrastructure. The VO maintains user membership information, while Sites control access and privileges to resources.

themselves [16]. The grid infrastructure components map such certificate attributes to accounts, ACLs, and other access controls on sites and at the boundaries to services, as well as apply policy -including black-and white-listing.

This security architecture implements a model of trust between sites and VOs with the grid organization. It relies on delegated trust between the VO and the end users, which must also be audited and assessed. Integration of newly emerging campus identification and authorization framework, Shibboleth [17] is underway as an extension to the existing X509 based infrastructures.

4.1.1. Lacks in the Security Infrastructure

Support for sub-VOs with inherited attributes and delegated trust is not fully available. Support for the extended attributes is restricted to a few services -notably processing, and soon storage.

Mechanisms for auditing and tracing of the end-to-end grid access and transactions are lacking.

Tools for tracing and assessing access across the whole ensemble of sites, VOs and services are not fully realised.

The safe storage and integrity of the certificates is a significant concern.

Delegation and use of proxy certificates is not fully supported by the end-to-end infrastructure.

4.2. Job Management and Execution

Sites present interfaces allowing remotely submitted jobs to be accepted, queued and executed locally. The priority and policies of execution are controllable by and at the site. Science driven infrastructures such as OSG and EGEE provide mechanisms for each research community to internally define priorities across particular groups and users and X509 certificate attribute extensions are used to support this.

Site policies and priorities are defined through mapping the user and attributes associated with their current transaction (see above) to specific accounts used to submit the job to a standard batch system. For example, OSG supports the Condor-G job submission client, which interfaces to either the pre-web service or web services GRAM Globus interface at the executing site. Job managers at the backend of the GRAM gatekeeper support job execution by local Condor, LSF, PBS, or SGE batch systems.

Of course, we expect that our distributed data center won't make the user have to select which remote system to submit their job to. Grid infrastructures offer a selection of automated resource selection services and meta-schedulers. The usefulness of such a service of course depends on the timeliness and accuracy of information about the state of the resources and their ability to successfully execute work that is sent to them. This is where the main challenges lie.

Many different such resource selectors exist. Using OSG as an example, there is support for stand-alone Condor match-making, a generalized Resource Selection Service from D0 and also interoperable support for the EGEE resource broker (RB) which provide user controllable mechanisms to automatically select to which site jobs are sent. Both RESS and the RB depend on the OSG site information services, which present information about the resources using the Glue Schema [17] attributes and providers and optionally converting the information to Condor Classads.

The user communities have realized that providing their own job management within the remote environment gives them more control over the prioritization and overall throughput of their applications. The largest science communities on OSG for example now use "pilot job" mechanisms. Pilot jobs are submitted through the normal job execution services. When the VO pilot job starts execution, using a standard batch slot, it interacts with its partner "VO job scheduler" to download the application executable to be run. The VO job scheduler controls the scheduling of jobs between the users in the VO and schedules jobs to run only on those resources that are immediately ready to execute them. These user jobs execute under the identity of the pilot job submitter↓, which can break the policy that sites must be able to identify the end user of their resources. OSG has integrated an Apache suexec [19] derivative, the EGEE glexec module, that enables the pilot to run jobs under the identity of the originating user.

Typical job submission and workflow usage is through user or community portals provided by the VOs themselves. These are increasingly sophisticated to hide the complexity and increase the automated throughput of the infrastructure for the indiviual user.

Scalability and response to overload are two challenges of the current job execution infrastructures. One major challenge that has yet to be well addressed is the reporting of well described and identified error conditions, and associated tools to allow easy user debugging of problems.

4.3. Data Transport, Storage and Access

Many of the early adopters using production grids have large file based data transport and application level high data I/O needs. The data transport, access, and storage implementations must take account of these needs. Implementation of file transport that follow the Globus GridFTP protocol for the raw transport of the data is ubiquitous. Invariably Globus GridFTP itself is used except where interfaces to storage management systems (rather than file systems) dictate individual implementations.

Several of the large infrastructures, including EGEE and OSG support the Storage Resource Management (SRM) [20] interface to storage resources that enables management of space and data transfers to prevent unexpected errors due to running out of space, overload of the GridFTP services, and to provide capabilities for pre-staging, pinning and retention of the data files. While there are several implementations of SRM OSG, for example, currently provides reference implementations of two storage systems -the LBNL Disk Resource Manager and dCache [21]. In addition, because functionalities to support space reservation and sharing are not yet available through grid interfaces, OSG defines a set of environmental variables (see Figure 5) that a site must implement and a VO can rely on to point them to available space, space shared between all nodes on a compute cluster, and for the use of high-performance I/O disk caches. These environment variables are used to distinguish between smaller, more long-lived storage for applications, and large, higher throughput, more transient areas for placement of data.

4.3.1. Challenges in Data Storage and Access

The major challenges for data storage and access are to provide implementations of managed storage areas that:

Are easy to install, configure and support;

Support the full range of disk storage resource sizes from several to hundreds of TeraBytes;

Provide guaranteed and opportunistic sharing of large amounts of space between and within VOs;

Deliver hundred terabyte data sets on-time;

Provide applications high throughput access to data at a local or remote site.

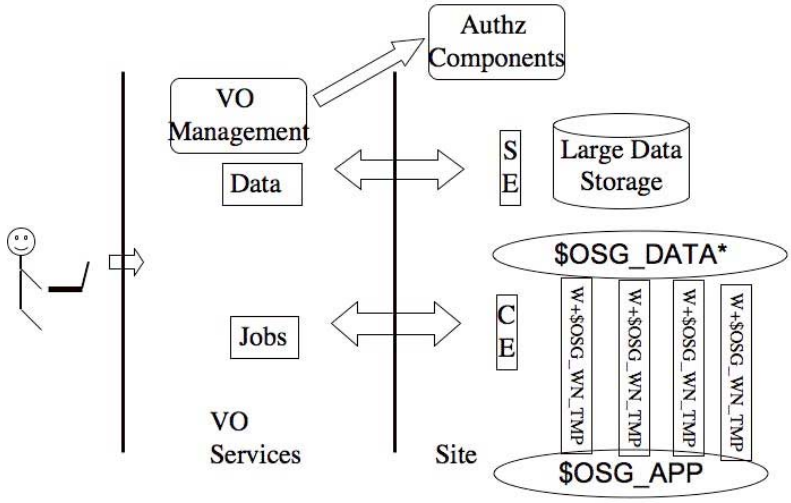


Figure 5: OSG Storage Environment Variables

4.4. Information Services

4.4.1. Service Status and Discovery

As mentioned above it must be accepted that information about the state and availability of the resources and services on a widely distributed and autonomously managed set of components will be incomplete and out of date. Reliance on this information must be cognizant of its quality and latency. A range of relatively static information can be used to dispatch work to sites, understand whether a configuration can support a particular application etc.

The idea of discovery of services and resources has long been a dream of transparent access to a distributed set of resources. While there have been some initial deployments a fully dynamic discovery service is not yet a reality. In general information is collected and checked against expected results and error conditions raised when the unexpected is noted.

4.4.2. Monitoring and Accounting

All production grids include, invariably many different, monitoring infrastructures. The VO, site and grid administrators and operations, the users and the management are all interested in many knowing the current and historical performance and load on the system. To enable discussion fo economics, accounting of the use of the resources is needed. This immediately brings us to an interesting question: If a resource is used it is clear whom to charge, but it may be that many different components have contributed to the value gained. If a job is scheduled through one grid infrastructure and executed through another – who accounts the job? Invariably, both do and this does indeed make sense.

One of the challenges of monitoring and accounting is to reduce the resource load due to the monitoring and accounting infrastructures themselves. The diversity and independence of the stakeholders in knowing the information, as well as the different views of to gather the information, make this a challenge in its own right.

4.5. Gateways to other Facilities and Grids

There is a rapid growth in the interest and deployment of shared computational infrastructures at the local and regional level. And in addition these is rapid growth in research communities' need to move data and jobs between heterogeneous facilities and build integrated community computational systems across high performance computing (HPC) facilities and more traditional computing clusters.

It was previously stated that the world is a "grid-of-grids" and interoperation and interfacing between them is part of the architecture and an operational goal. For the OSG the scope includes software and support for campus and regional organizations to form their own locally shared facilities and distributed infrastructures which gateway to the national grid. Three examples participating in OSG today are the Fermilab Facility Grid (FermiGrid) [22], the Grid Laboratory of Wisconsin (GLOW)[23] and the Purdue University campus-wide infrastructure. Each local organization contributes to and manages the gateway between the local facility and the OSG. The implementation principles of the gateways for the first two differs:

On FermiGrid each VO interfaces their own resources directly to the OSG and then accesses them through their regular remote access mechanisms. FermiGrid interfaces a gateway to the OSG, which dispatches jobs from all other OSG VOs automatically across the complete set of resources (see Figure 6).

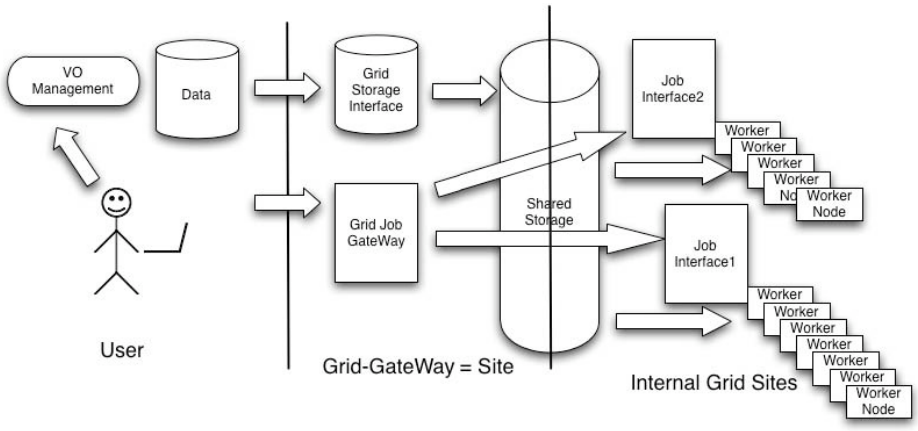


Figure 6: Access between Grids (example OSG -> FermiGrid).

GLOW provides a single interface to the OSG. Local job and data submissions are handled through the existing local mechanisms and then automatically "uploaded" to OSG resources under control of the GLOW gateway.

The campus infrastructures provide multiple access mechanisms to the processing nodes so that computing cluster appear as local Condor pools as well as being accessible by OSG and, in the case of Purdue, TeraGrid.

Also as stated above, the large production grids federate and supporting groups to submit jobs across and move data between them. For example, the OSG collects information from the resources and publishes them in the format needed by the EGEE. A VO "Resource Broker" or job dispatcher can then submit jobs transparently across resources on the two grids.

5. Managing the Production Grid

Treating a production grid as a data center brings new insight and requirements into what is required to manage and operate them. Configuration management, service level management, dynamic resource control and so forth together with the life cycles of and relationships between the components become important to sustain a robust and evolutionary infrastructure. There are some unique challenges that the remoteness and autonomy of the components of the infrastructure bring forth.

Attention must also be paid to the provisioning and lifecycle of the components and thought given to what information is needed from all components, how to deal with a system where some components are always non-operational at any one time, how to reason given the latency in the information available, how to reconcile the accounting by different parties accessing information at different layers of the system, and how a uniform management can be layered onto a set of disparate entities not under direct control of the system itself. Such management includes monitoring as well as managing the state transitions of the services (autonomous in the OSG model). Relationships and dependencies need to be understood also. These can be described as objectives for the level of service

(Service Level Objectives) which defines the goals for the components attributes in terms of up time, defense from overload, replicability etc. Measurement, control and life-cycle management of grid service components then become part of an overall management infrastructure [24], with a set of defined states and interfaces of its own and into which can be plugged new services as they are deployed.

6. Challenges for the Future

Today's production research and science grids are charting new territories in several areas: Community driven production support and evolution of large complex systems with coordination but no control over the "end-points"; Security and operations across autonomously owned and managed facilities which scale from small university department clusters to large leadership class high performance computing facilities; Transparent support and failover during update of software and services; Robustness against failure, overload and unintentional and intentional misuse; Development of a sustainable economic model for use and growth of the infrastructure.

Not just the number of jobs executed and the amount of data transferred and stored must measure success. Success has to be measured by the impact on scientific productivity and maturity of computation as a cornerstone, together with experiment and simulation, of the research portfolio. While many measurements are made and much data is being collected to help us determine the impact, we do not yet really understand how to translate and analyze this information to quantify value and benefit.

Acknowledgments

Operational production quality grids are due to the intellectual commitment and hard work of many people. The OSG Consortium, staff and collaborators are a wonderful team who has made the OSG implementation possible. This work is supported by the Office of Science, U.S. Department of Energy, SciDAC program under Contract DE-FC02-06ER41436 and the National Science Foundation Cooperative Agreement, PHY-0621704.

References

[1] Enabling Grids for EsciencE (EGEE) web site at http://www.eu-egee.org/
[2] GridPP home page at http://map.gridpp.ac.uk/
[3] NDGF home page at http://www.ndgf.org/index.php?page=about
[4] Open Science Grid home page at http://www.opensciencegrid.org.
[5] TeraGrid home page at http://www.teragrid.org/
[6] LHC home page: http://lhc.web.cern.ch/lhc/
[7] LIGO http://www.ligo.caltech.edu/
[8] Definition of Virtual Organization: http://en.wikipedia.org/wiki/Virtual_organization
[9] OSG Blueprint http://osg-docdb.opensciencegrid.org/0000/000018/005/OSG-Blueprintv0.10.pdf
[10] The Virtual Data Toolkit web site: http://vdt.cs.wisc.edu
[11] Condor ™project web site: www.cs.wisc.edu/condor
[12] Globus Tookit™ web site: www.globus.org/toolkit
[13] Metronome (formerly NMI build and test): http://nmi.cs.wisc.edu/
[14] OSG Security plan: http://osg-docdb.opensciencegrid.org/0003/000389/017/OSGSecurityPlan.doc
[15] International Grid Trust Federation: http://www.gridpma.org/

[16] Virtual Organization Management systems for EGEE and OSG: R. Alfieri, R. Cecchini et al: Managing Dynamic User Communities in a Grid of Autonomous Resources, Talk from the 2003 Computing in High Energy and Nuclear Physics (CHEP03), http://osg-docdb.opensciencegrid.org/0006/000623/001/Certificate%20Attribute%201.0.doc
[17] Shibboleth federated Single-SignOn and attribute exchange framework: http://shibboleth.internet2.edu/
[18] Glue Schema Open Grid Forum Working Group: http://forge.gridforum.org/sf/projects/glue-wg
[19] Apache suexec: http://httpd.apache.org/docs/1.3/suexec.html
[20] Storage Resource Management Collaboration: http://sdm.lbl.gov/indexproj.php?ProjectID=SRM
[21] dCache Consortium web site: www.dcache.org
[22] FermiGrid web site: http://fermigrid.fnal.gov
[23] Grid Laboratory of Wisconsin web site: www.cs.wisc.edu/condor/glow.
[24] EGA reference model: http://tinyurl.com/3d942b.

This page intentionally left blank

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Subject Index

High Performance Computing and Grids in Action
L. Grandinetti (Ed.)
IOS Press, 2008

Author Index

This page intentionally left blank