RISC-Linz

Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria, Europe

3. Austrian Grid Symposium

Jens VOLKERT, Wolfgang SCHREINER,
Thomas FAHRINGER (eds.)

Johannes Kepler University Linz, Austria
September 28-29, 2009

RISC-Linz Report Series No. 09-14

Editors: RISC-Linz Faculty
B. Buchberger, R. Hemmecke, T. Jebelean, E. Kartaschova, M. Kauers, T. Kutsia, G. Landsmann, F. Lichtenberger, P. Paule, N. Popov, H. Rolletschek, J. Schicho, C. Schneider, W. Schreiner, W. Windsteiger, F. Winkler.

Supported by: BWMF (Austrian Federal Ministry of Science and Research) contract GZ BMWF-10.220/0002-II/10/2007 (Austrian Grid).

3rd Austrian Grid Symposium, September 28th-29th, 2009

The Austrian Grid is the national Grid initiative in Austria to support grid computing in general, and to provide coordination and collaboration between research areas actively involved and interested in grid computing. The Austrian Grid Consortium combines Austria's leading researchers in advanced computing technologies with well-recognized partners in grid-dependent application areas.

This is the third symposium organised by the Austrian Grid partners to demonstrate results of the cooperation (earlier symposia have been held in Hagenberg and Innsbruck). The symposium will be a forum for the institutions deploying the Austian grid infrastructure, for the Austrian developers of grid middleware extensions, and for the users of the Austrian Grid infrastructure to meet each other, share ideas and experience, and discuss future directions of the national grid efforts. Moreover, the symposium will gather an international research community that works on the development and application of state of the art Grid technology and reports on experience with Grid applications.

Main topics of the symposium will be as follows:
• Grid infrastructure deployment and management
• Grid security and authentication services
• Grid middleware services
• Scientific visualisation on the grid
• Grid user interfaces and web portals
• Multimedia encoding and transportation in the grid
• Data base access and data mining on the grid
• Grid programming environments
• Grid-enabled applications and grid-dependent application domains

3rd Austrian Grid Symposium, September 28th – 29th, 2009
Johannes Kepler University Linz, Austria

Day 1	***3rd Austrian Grid Symposium***	***28.09.09***
12:30-13:00	Symposium Opening	
13:00-14:00	Invited Talk: Clouds-New Services or only Hype?	Prof. Dr. Dieter Kranzlmüller, Ludwig Maximilian University Munich, Leibniz Supercomputing Center
14:00-14:30	Coffee Break	

Session 1	**Uri-Center Rep.-R C**	**Author**
14:30-14:55	The Swiss National Grid Association and its Experience on a National Grid Infrastructure	Nabil Abdennadher, Dean Flanders, Sigve Haug, Pierre Kuonen, Heinz Stockinger, Christoph Witzig
14:55-15:20	On Geo-location Based Data Replication for Mobile Grids	Harald Meyer, Karin Anna Hummel
15:20-15:45	A Supercomputing API for the Grid	Karoly Bosa, Wolfgang Schreiner
15:45-16:10	Analysing a petabyte-scale distributed data management system	Mario Lassnig, Vincent Garonne, Angelos Molfetas, Miguel Branco
16:10-16:30	Coffee Break	

Session 2	**Uni-Center Rep.-R C**	**28.09.09**
16:30-16:55	Research on Security Requirements for Grid Environments	David Huemer, A Min Tjoa, Marco Descher
16:55-17:20	A RESTful Repository to Store Services for Simple Workflows	Ralph Vigne, Jürgen Mangler, Erich Schikuta, Christoph Witzany
17:20-17:45	Grid Gateway-Accessing grid resources from private networks	Stephan Leitner, Michael Krieger
17:45-18:10	GRID-based Motion tracking of the Heart Wall	Bernhard Quatember, Martin Mayr, Wolfgang Recheis, Stefanos Demertzis, Giampietro Allasia, Roberto Cavoretto, Alessandra De Rossi, Ezio Venturino
20:00:00	Conference Dinner	Arkadenhof, Landstrasse 12, 4020 Linz

Austrian Grid is a project funded by the BMWF (Federal Ministry of Science and Research). Contract: GZ BMWF-10.220/0002-II/10/2007

Day 2	***3rd Austrian Grid Symposium***	***29.09.09***
09:00-10:00	Invited Talk: An Overview of CRO NGI Uni-Center Rep.-R C	Dobrisa Dobrenic University Computing Center-SRCE, University of Zagreb
10:00-11:40	Industry Session, Uni-Center, Sitzungszimmer 2	

Session 3	Uni-Center Rep.-R C	Author
10:00-10:25	Assuring Grid Access in the g-Eclipse Project	Jie Tao, Matthias Stümpert
10:25-10:50	Token Based Authentication Methods for the Grid	Thomas Ludescher, Willy Weisz, Marco Descher
10:50-11:15	Software Execution Constraints for Correctness Checking of Workflows	Maximilian Berger, Romedius Weiss, Thomas Fahringer
11:15-11:40	Providing Semantic Data Integration as a Grid Service	Thomas Leitner, Andreas Langegger, Wolfram Wöß
11:40-13:00	Lunch Break	

Session 4	Uni-Center Rep.-R C	Author
13:00-13:25	Ray Tracing in Grid Environments	Thomas Odaker, Jens Volkert
13:25-13:50	Developing Astrophysics Workflow Applications with the ASKALON Environment in the Austrian Grid	Simon Ostermann, Markus Brejla, Radu Prodan, Thomas Fahringer, Sabine Schindler
13:50-14:15	Running Hierarchical N-Body Simulations on Grid Infrastructures	Paul Heinzlreiter, Jens Volkert
14:15-14:40	Gridication of the Solution of Highdimensional Improper Integration and Integral Equations	Peter Zinterhof sen., Peter Zinterhof jun.
14:40-15:00	Coffee Break	

Austrian Grid is a project funded by the BMWF (Federal Ministry of Science and Research). Contract: GZ BMWF-10.220/0002-II/10/2007

3rd Austrian Grid Symposium, September 28th – 29th, 2009
Johannes Kepler University Linz, Austria

Session 5	Uni-Center Rep.-R C	Author
15:00-15:25	Prediction of Precipitation in the Alps using ASKALON on the Austrian Grid	Kassian Plankensteiner, Johannes Vergeiner, Radu Prodan, Georg Mayr, Thomas Fahringer
15:25-15:50	Large Scale Data Analysis on the Grid in the Search for New Physics	Dietrich Liko
15:50-16:15	Austrian Federated WLCG Tier-2	Peter Oettl, Gregor Mair, Reinhard Bischof, Gerhard Walzel, Natascha Hoermann
16:15-16:40	Network Based Execution of Dynamic Workflows in Grid and Cloud Based Environments	Gerhard Stürmer, Jürgen Mangler, Erich Schikuta

Session 6	Uni-Center Rep.-R C	Author
16:50-17:15	A Gridenabled Solver for the Fluidstructure Interaction (FSI) Problem	Walter Zulehner, Ulrich Langer, Huidong Yang
17:15-17:40	Parallel Algebraic Multigrid on General Purpose GPUs	Gundolf Haase, Manfred Liebmann, Gernot Plank, Craig Douglas
17:40-18:05	Secure Scalable Video Compression for Gvid	Heinz Hofbauer, Thomas Stütz, Andreas Uhl
18:05-18:30	Grid-based Scientific Dataspace Support Platform for Breath Gas Analysis	Ibrahim Elsayed, Thomas Ludescher, Philip Masser, Thomas Feilhauer, Peter Brezany

Austrian Grid is a project funded by the BMWF (Federal Ministry of Science and Research). Contract: GZ BMWF-10.220/0002-II/10/2007

Committee

General Chair:

- Jens Volkert (GUP, Joh. Kepler University Linz, Austria)

Honorary Chairmen:

- Bruno Buchberger (RISC, Joh. Kepler University Linz, Austria)
- Dietmar Kuhn (University of Innsbruck, Austria)

Program Chairmen:

- Thomas Fahringer (IFI, University of Innsbruck, Austria)
- Wolfgang Schreiner (RISC, Joh. Kepler University Linz, Austria)

Local Chairmen:

- Rene Kobler (GUP, Joh. Kepler University Linz, Austria)

Program Committee:

- Sigi Benkner (University of Vienna, Austria)
- Peter Brezany (University of Vienna, Austria)
- Marian Bubak (AGH Krakow PL and UvA Amsterdam NL)
- Christian Glasner (Johannes Kepler University, Austria)
- Guenter Haring (University of Vienna, Austria)
- Ladislav Hluchy (Slovak Academy of Sciences, Slovakia)
- Karin Hummel (University of Vienna, Austria)
- Wolfgang Karl (University of Karlsruhe, Germany)
- Jacek Kitowski (AGH Krakow, Poland)
- Rene Kobler (Johannes Kepler University, Austria)
- Harald Kornmayer (NEC Laboratories Europe, Germay)
- Gabriele Kotsis (Johannes Kepler University, Austria)
- Dieter Kranzlmueller (LMU Munich, Germany)
- Erwin Laure (KTH Royal Institute of Technology, Sweden)
- Laurent Lefevre (INRIA/LIP, France)
- Robert Lovas (MTA Sztaki, Hungary)
- Marcin Paprzycki (Warsaw School of Social Psychology, Poland)
- Erich Schikuta (University of Vienna, Austria)
- Wolfgang Schreiner (Johannes Kepler University, Austria)
- Achim Streit (FZ Juelich, Germany)
- Jie Tao (Forschungszentrum Karlsruhe, Germany)
- A Min Tjoa (University of Vienna, Austria)
- Andreas Uhl (COSY, University of Salzburg, Austria)
- Willy Weisz (University of Vienna, Austria)
- Roland Wiesmueller (University of Siegen, Germany)
- Wolfram Woesz (Johannes Kepler University, Austria)

Invited Talk 1:

Cloud-New Services or only Hype?

Abstract
The Cloud is often referred to as the next big thing in distributed computing. Cloud providers offer different kinds and different levels of services, leading to new kinds of application and raising interest from many, including the media. But what is really behind this new approach of using networking resources for community applications? This talk intends to provide a neutral survey of cloud solutions with an in-depth analysis of specific characteristics. The starting point is a classification of cloud services, explaining the differences and available solutions. Experiments with actual clouds provide more detailed information about the viability of using the cloud for science and research, and offer a glimpse on today's benefits and future drawbacks of cloud computing.

Prof. Dr. Dieter Kranzlmüller
Ludwig-Maximilian University Munich and Leibniz Supercomputing Centre

Invited Talk 2:

An Overview of CRO grid NGI

Abstract
CRO NGI (Croatian National Grid Infrastructure) is an integrated allocated computer environment, consisting primarily of computer (processing) and data (disc and tape) resources, which are located in geographically allocated sites within the Republic of Croatia. CRO NGI is a common resource of the scientific and academic community and represents the fundamental infrastructure for the scientific research, the application of new technologies and the integration of Croatia and Croatian scientists into the European Research (ERA) and European Higher Education (EHEA) Area.
The Coordinator of CRO NGI is the University Computing Centre University of Zagreb - SRCE. The CRO NGI Board, appointed by the minister responsible for science, the Council of Partners and the Council of Users take part in the management. CRO NGI is financed as a separate unit in the State Budget of the Republic of Croatia. The first CRO NGI Partner Contracts were signed on November 5, 2007 by four institutions, plus two Partners by default - the Ministry of Science and Education and CARNet, the Croatian NREN.
The Croatian National Grid Initiative (CRO-GRID) is a voluntary-based association of academic and research institutions, government institutions and commercial companies gathered around CRO NGI. Founders and members of CRO-GRID are the institutions which took part in the poly-project CRO-GRID from 2004 to 2007.

Dobrisa Dobrenic
University Computing Center-SRCE, University of Zagreb

THE SWISS NATIONAL GRID ASSOCIATION AND ITS EXPERIENCE ON A NATIONAL GRID INFRASTRUCTURE

Nabil Abdennadher[1], Dean Flanders[2], Sigve Haug[3], Pierre Kuonen[4], Heinz Stockinger[5], Christoph Witzig[6]

Abstract

In the following article we provide our experience with the creation of the Swiss National Grid Association (SwiNG) as well as the establishment of a nation-wide Grid infrastructure based on the ARC Grid middleware. Although not yet fully in production, we have already several scientific user communities in different domains (high energy physics, snow and avalanche related physics, biochemistry, proteomics (bioinformatics) and computer security).

1. Introduction

In the last decade, there have been many national and international Grid middleware and infrastructure projects to mainly support scientific applications in various domains such as physics or life sciences; only recently we see real industry involvement in terms of cloud computing. The main Grid middleware systems in Europe are gLite (developed and deployed via the EGEE project, http://www.glite.org), ARC (developed and deployed via the NorduGrid collaboration, http://www.nordugrid.org) as well as UNICORE (deployed in DEISA, http://www.unicore.eu). Many of the middleware development, maintenance and operation costs are in part covered by funding from the European Commission via the different framework programmes or via national funds. Since Grids are considered to be a mature technology by now, various funding agencies want to "harvest" what they have promoted and supported in the last decade, and see stable Grid infrastructures. What is more, scientists and Grid practitioners request a sustainable infrastructure with certain guarantees beyond the life times of typical 2 to 4 year projects. As a consequence, more than 30 European countries are involved in the design study for a European Grid Initiative (http://www.eu-egi.org) which explores the possibilities to establish such a European, sustainable Grid infrastructure. A similar model is already in place for research networks, where each country provides a single contact point (http://www.terena.org).

1 University of Applied Sciences Western Switzerland, HEPIA, Geneva, Switzerland
2 Friedrich Mischer Institute, Basel, Switzerland
3 University of Berne, Lab. for High Energy Physics, Berne, Switzerland
4 University of Applied Sciences Western Switzerland, HEFR, Fribourg, Switzerland
5 Swiss Insitute of Bioinformatics, Vital-IT Group, Lausanne, Switzerland
6 SWITCH, Zurich, Switzerland

Following the requirement to provide a single National Grid Initiative per country, the Swiss National Grid Association (SwiNG) was officially created in May 2007 by academic and scientific organisations in Switzerland. SwiNG is a legal entity that represents the interests of the Swiss Grid community and co-ordinates a Grid infrastructure. In the following article we provide our experience with the creation of the association as well as the establishment of a nation-wide Grid infrastructure based on the ARC Grid middleware. Currently, our scientific user communities are from several different domains (high energy physics, snow and avalanche related physics, biochemistry, proteomics (bioinformatics) and computer security). The Grid infrastructure is partially used in production mode by individual projects and research domains.

2. The Swiss National Grid Association (SwiNG)

Although the official efforts to create a Swiss National Grid Initiative have only started in 2006, the first multi-site Grid infrastructures had been established already. Within the bioinformatics community the Swiss Bio Grid project [3] connected several bioinformatics HPC centres and the Swiss National Supercomputing Centre using ARC. The project ran officially from spring 2004 until December 2006. This early experience dedicated to life sciences relied on best-effort and volunteers from different sites since no infrastructure nor maintenance funding was available. This is in contrast with projects such as EGEE where sites usually receive funds to maintain and operate gLite. A second early multi-site infrastructure is the Swiss ATLAS Grid [2]. Since 2005 it operates four ARC connected clusters in Bern, Geneva and Manno (CSCS). Consequently, the efforts within the Swiss Bio Grid and the Swiss ATLAS Grid can be seen as a first Grid success in Switzerland with respect to infrastructure provision.

Based on the above experience as well as on participation in other national and international projects of various Swiss partners (CoreGrid, EGEE, EMBRACE, KnowARC, etc.), the work for a national Grid initiative started with the aim to support various scientific domains in Switzerland. One of the first tasks was to provide a prove-of-concept Grid infrastructure that goes beyond one scientific domain. This was achieved via the so called “seed project” [1]. In May 2007, the Swiss National Grid Association was formally created as a legal entity (http://www.swing-grid.ch). The association is guided by the following mission:

1. Ensure competitiveness of Swiss science, education and industry by creating value through resource sharing.
2. Establish and coordinate a sustainable Swiss Grid infrastructure as a dynamic network of resources across different locations and administrative domains.
3. Provide a platform for interdisciplinary collaboration to leverage the Swiss Grid activities, supporting end-users, researchers, education centers, resource providers and industry.
4. Represent the interests of the national Grid community towards other national and international bodies.

The association is primarily dedicated to research, i.e. scientific and academic institutions and their applications. Currently, 19 institutions (i.e. almost all Swiss universities, federal institutes of technology, universities of applied sciences as well as specialised research institutes) are among the members. This shows that there is considerable interest in Grid computing in Switzerland.

In order to attract scientists to use Grid computing, the association has the mandate to establish and co-ordinate the implementation, maintenance and operation of a nation-wide Grid infrastructure. It is important to note that SwiNG does not own the hardware infrastructure itself but it stays property of the individual institutional members. A recently funded project called the SMSCG (see Chapter 3) has been started to establish such an infrastructure and to support a small set of pilot applications.

3. Swiss Multi-Science Computing Grid Project (SMSCG)

The primary goal of the SMSCG project (http://www.smscg.ch) is to provide computational resources to solve scientific computational problems. This involves the installation, commissioning, and operation of a computational Grid across several institutions of the Swiss higher education sector with active (user driven) involvement of applications from different scientific domains. Currently, the following scientific applications are supported and pre-installed at several sites:

1. ATLAS (High Energy Physics application developed for the LHC experiment at CERN)
2. NAMD and GROMACS (biochemistry applications)
3. POP-C++ (Parallel Object Programming framework)
4. Alpine 3D (an application for the high resolution simulation of alpine surface processes)
5. swissPIT (Swiss Protein Identification Toolbox)
6. JOpera (open grid workflow management system based on the Eclipse platform)
7. RSA768 (computer security/cryptography application)

One can certainly also run other applications that are available on local clusters (e.g. a large set of life science applications is available on Vital-IT) or one can submit own applications either in source or binary format (SMSCG supports various flavours of Linux such as Scientific Linux CERN, CentOS, etc.). Sites that participate in SMSCG use ARC 0.6 as the main middleware toolkit to allow for secure inter-site job submission and efficient data transfer. The actual sites involved in the current infrastructure are depicted in Figure 1. Overall, in the order of 2500 processing cores are available to Swiss researchers.

By default, ARC supports GSI, i.e. every user is authenticated via an X.509 user certificate. Switzerland already uses Shibboleth at all universities which allows for a national authentication system. Additionally, the Short Lived Credential Service (SLCS) [4] can be used together with

Shibboleth to request SLCS X.509 certificates. These certificates work with ARC and avoid that users have to explicitly request X.509 certificates, a considerable burden for non IT specialists.

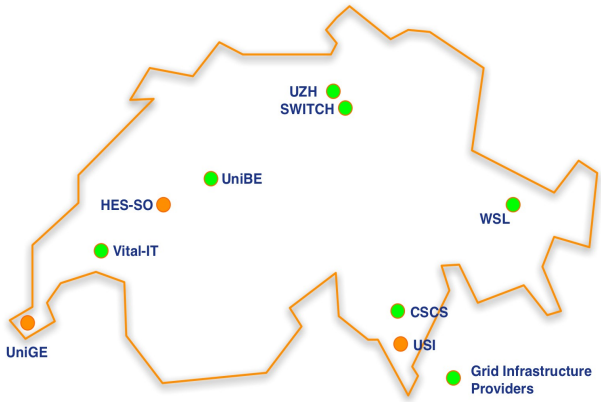


Figure 1. **Sites involved in the Swiss Multi-Science Computing Grid** (cf. http://www.smscg.ch for abbreviations)

4. Conclusion

The foundation of the Swiss National Grid Association has created a new momentum in Grid computing in Switzerland. The SMSCG project is only one of the new projects that are currently active. Additionally, there are related projects that build on desktop Grid infrastructures which will partly be included in the national Grid infrastructure. Examples for such desktop Grids are the UNIL-Grid (at the University of Lausanne), the Greedy project at the EPFL (Lausanne), the VirtualEZ Grid (lead by HES-SO/HEPIA Geneva to connect different desktop Grids) as well as projects in Berne and Zurich.

References

[1] ABDENNADHER, N. et al. Initializing a National Grid Infrastructure - Lessons Learned from the Swiss National Grid Association Seed Project. *8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid2008)*, IEEE Computer Society Press, Lyon, France, May 19-22, 2008.

[2] COGNERAS, E. et. al., The Swiss ATLAS Grid, *4th International Conference on Grid and Pervasive Computing (GPC'09)*, Springer Verlag, Geneva, Switzerland, May 4-8, 2009.

[3] PODVINEC, M. et al. The SwissBioGrid Project: Objectives, Preliminary Results and Lessons Learned. *2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006) - Workshop on Production Grids*, IEEE Computer Society Press, Amsterdam, The Netherlands, Dec. 4-6, 2006.

[4] SLCS: http://www.switch.ch/grid/slcs/, May 2009.

On Geo-location Based Data Replication for Mobile Grids – Extended Abstract

Harald Meyer* and Karin Anna Hummel*

1. Introduction and Related Work

Mobile devices can be integrated into grids not only for accessing grid resources but also to provide resources, like reliable storage. Among the most important challenges in mobile distributed networks are limited bandwidth of wireless connections and varying availability of resources [6]. Replication of data is a well-known means to address these challenges [2, 5]. Another important issue for mobile distributed systems is addressing and keeping track of the identity and address of a communication partner (i.e., IP address and port number). Solutions like mobile IP are often introducing high overhead. Hence, new concepts are required for better performing mobility management of nodes which change their (geo-) locations frequently.

For mobile data grids, we introduce an information and replication layer reflecting these considerations by using geo-location information to best place and manage data. Hereby, we focus on replication to increase availability while keeping the number of replicas reasonably low. The approach is dedicated to data related to a geographic location, e.g., the data is only of interest within a particular area limiting the range of dissemination [1, 3], and assumes that mobile nodes are aware of their location, e.g., by using GPS data. Around this location, a virtual geometric structure determining *special forwarding locations* for optimal coverage of the area is defined which is common knowledge among nodes. Nodes in proximity of these special locations take actively part in the distributed replication algorithm (replica creation, replica update, and voting for consistency) while others remain passive. Related approaches limit the number of replicas by considering the network topology around a point of interest and storing replicas only on each n-th hop [7]. Such approaches suffer from node mobility, where the expected benefit of structuring decreases. In contrast, our approach can maintain the structure even in mobile scenarios. Hence, we expect resulting performance benefits in terms of lower numbers of replicas and decreased communication overhead. In the remainder of this paper we present the geo-location based replication approach in Section 2 and conclude in Section 3.

2. Location Based Data Replication

Following the vision of information bound to geo-locations (and not to computers and computer addresses), our geo-location based data replication approach aims at efficient data storage in mobile grids. The distributed algorithm makes data available in dynamic mobile environments and supports consistent replica updates. To make data available in areas of interest, data is replicated among mobile nodes residing in this area up to a configurable range. This area is termed *Region of Interest* (RoI)

*Department of Distributed and Multimedia Systems, University of Vienna, Lenaugasse 2/8, A-1080 Vienna, email:{harald.meyer|karin.hummel}@univie.ac.at

defined by a circle around the *Point of Interest* (PoI). By selecting nodes positioned best in the RoI for storing data and, thus, making data available, efficient data replication is possible in terms of reduced number of replicas and reduced messaging overhead.

2.1. Basic Replica Placement Approach

The replication process follows the concept of decentralized opportunistic data replication where nodes autonomously decide how they want to take part in the replication process. Upon receiving a data item, each node makes a local copy and estimates its geo-position retrieved, e.g., from a GPS receiver in outdoor scenarios. Additionally, each node knows the geo-structure of the RoI and decides locally whether it is in a good position to replicate (forward) the data further. Hereby, nodes are assumed to be able to communicate with neighboring nodes over a wireless network (like IEEE 802.11 ad-hoc mode) and to hold local scan information about the nodes currently in the surrounding area to estimate the current node density.

In Figure 1(a), an overview of the concept is depicted in a 2D area. Each data item i is related to a fixed Cartesian location $l = (x, y)$. Data items are of interest for users in the RoI by a circle of radius r_l around the data item's location l. Data items are replicated only within the radius r_l.

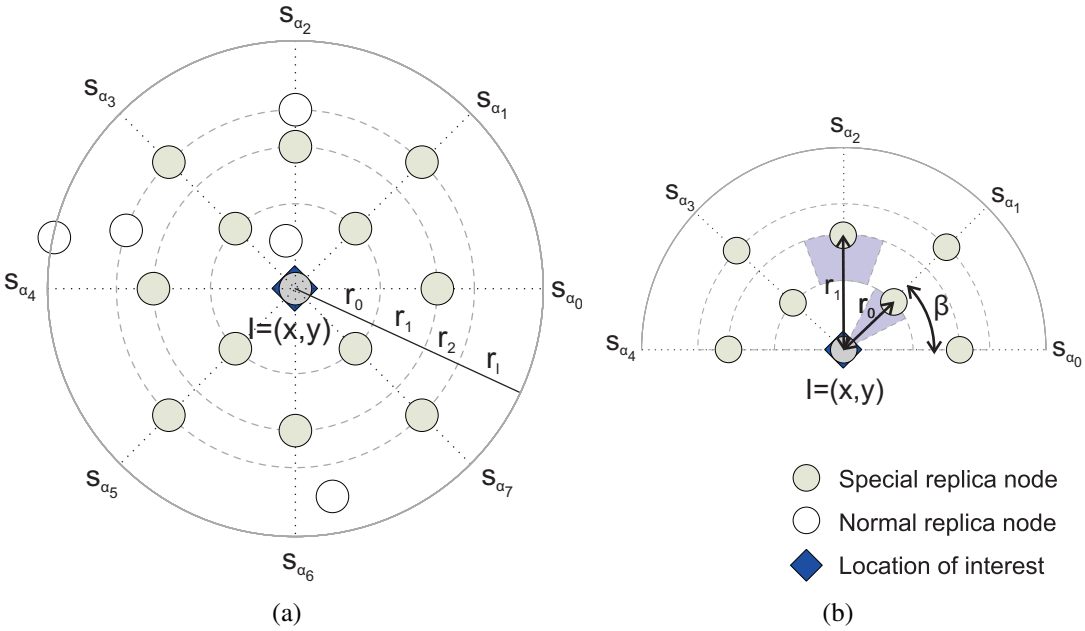


Figure 1. (a) Overview of the location-dependent replication approach and (b) detailed view of the concept.

Due to limited resources of mobile nodes and limited bandwidth of wireless connections [6], data forwarding by all nodes in the network can drain resources and overload the network. Furthermore, disconnections can lead to partitioning of the network and inconsistencies among replicas. Our approach addresses these issues by controlling data replication and dissemination in two ways. First, data items are only replicated within their RoI which limits replication overhead to the RoI and lowers the probability of partitioning. Second, communication overhead is reduced by separating active *special nodes* R_s from passive *normal nodes* R_n. When an R_n node receives a data item, it just stores it locally, whereas an R_s node also forwards (replicates) the data item to its neighbors. Additionally, only R_s nodes take part in the voting process to achieve consistency. The distinction between R_s and

R_n nodes is made depending on their locations. Hereby, each node decides upon its type locally and autonomously without additional communication overhead.

A node requesting a data item broadcasts a message with the item's name or a location associated with the item. The message is assigned a unique ID (unique node ID and a local node counter). The message is forwarded via broadcast until a maximum hop count or a node which holds a replica of the item is reached. If the data item is found, it is transfered back via the inverse request route.

2.2. On Structuring the Region of Interest

Based on its position and local node density, each node autonomously decides if it wants to become a special node. Therefore, the RoI is divided into n sectors $s_{\alpha_{j=0}}, ...s_{\alpha_{j=n-1}}$ of equal size $\beta = \frac{360}{n}$ and m concentric circles $r_{i=0}, ...r_{i=m-1}$ with radii of different size, where the difference between radii decreases with increasing distance to the center of the RoI, so that transmission ranges of optimal placed nodes allow intersections. Optimal forwarding positions are defined in the center of the RoI and on specific intersection points between section borders s_{α_j} and circles r_i. For circles r_i with $i \bmod 2 = 0$ optimal positions are intersections between r_i and every second arc border, starting with the arc with angle β, 3β, 5β, etc. Circles with $i \bmod 2 = 1$ have optimal intersection positions every second circle sector border starting with the sector with angle 0, β, 2β, etc. By knowing these optimal intersection positions and their current positions, nodes decide to become special nodes if they reside at such an intersection. In Figure 1(a), a sample structure for $n = 8$, $\beta = 45$ with three radii r_0, r_1, and r_2 is depicted.

In Figure 1(b), parts of the RoI are presented which show the intersections of sector borders and circles, special nodes residing at these intersections, and shaped areas in proximity of the intersection. These shaped areas visualize regions which enlarge the set of optimal positions depending on the currently and locally measured node density. This is necessary, since nodes will not frequently reside exactly at each optimal position, but rather in proximity to these positions. The area is defined by the section borders $s_{\alpha_j-\beta}$ and $s_{\alpha_j+\beta}$ and circle radii r_{i-1} and r_i (for $i = 0$ the area is a sector limited by the center of the RoI and r_0). The whole area is used in case of minimal node density. With increasing node density, the area is reduced proportionally. Hereby, it is heuristically assumed that the likelihood of a node near the optimal position is higher for higher node densities.[1]

2.3. Data Creation, Updates, and Consistency

Whenever a data item is created it is disseminated within its RoI. First the node which created the data item broadcasts the item together with a unique message ID to its neighbors within transmission range. Nodes which are within the RoI (the replication radius r_l is stored in the header of the message) make a local copy if they have sufficient resources. Nodes on optimal replica positions forward the data item message via broadcast. If a message with the same unique message ID arrives multiple times at a node, only the first message is forwarded and subsequent messages are ignored.

For data updates and consistency, our first approach is based on additional messaging among R_s nodes. Data updates are performed based on a hierarchical voting protocol modeling neighborhood relations in hierarchies which achieves eventual consistency. First, the node which wants to update an item broadcasts an update requests, which is disseminated among R_s nodes in the RoI, and waits

[1]Note, that an uniform distribution of nodes in the sectors is assumed by this heuristic. If this assumption is not fulfilled, the benefits of the algorithm cannot be fully exploited.

for answers. Each R_s node makes a vote for an update if its local data item's version number is lower than the update's version number, and all its neighbors which are farther away from the PoI voted for the update. The waiting time for votes decreases with the radius in the RoI. Nodes at the perimeter of the RoI do not wait for votes, while the timeout for nodes closest to the PoI is highest. The requesting node collects all votes and broadcasts the update if the majority of answers votes for the update.

2.4. Evaluation

The proposed algorithm has been implemented and evaluated by means of discrete event simulation using OMNet++ [8] and the INET framework. First results in an experimental area of 500 to 500 meters with varying node density and using the Manhattan mobility model show that the algorithm allows to reduce replica creation messaging overhead significantly when compared to flooding based replication up to 92% and for replica updates up to 72% (high node density of 40 nodes in the area). More details on this first evaluation can be found in [4].

3. Conclusions

Based on the assumption that data is related to geographic locations in mobile grids, we proposed a geo-location based replication protocol which allows to keep data available around a point of interest (within a region of interest) in an opportunistic mobile grid. By considering a geometric structure and local node densities, nodes decide autonomously whether they are passive replicators or whether they reside in optimal positions to participate in further replicating (forwarding) the data, updating, and executing a hierarchical voting protocol for replica consistency. First encouraging results show that considering the geo-location for replication decisions allows to reduce messaging overhead by achieving similar availability as a pure flooding based replication approach. In future extended work, we will enhance and detail our investigations for efficient updates and consistency.

References

[1] Margaret H. Dunham and Vijay Kumar. Location Dependent Data and its Management in Mobile Databases. In *DEXA '98: 9th International Workshop on Database and Expert Systems Applications*, pages 414–419, 1998.

[2] Abdelsalam A. Helal, Bharat K. Bhargava, and Abdelsalam A. Heddaya. *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers, 1996.

[3] Dik Lun Lee, Jianliang Xu, Baihua Zheng, and Wang-Chien Lee. Data Management in Location-Dependent Information Services. *IEEE Pervasive Computing*, pages 65–72, 2002.

[4] Harald Meyer and Karin Anna Hummel. A Geo-location Based Opportunistic Data Dissemination Approach for MANETs. In *CHANTS'09: 4th ACM Workshop on Challenged Networks (accepted for publication)*, 2009.

[5] Yasushi Saito and Marc Shapiro. Optimistic Replication. *ACM Computing Surveys*, pages 42–81, 2005.

[6] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *15th Annual ACM Symposium on Principles of Distributed Computing*, pages 1–7, 1996.

[7] Masahiro Tamori, Susumu Ishihara, Takashi Watanabe, and Tadanori Mizuno. A Replica Distribution Method with Consideration of the Positions of Mobile Hosts on Wireless Ad-Hoc Networks. In *ICDCS '02: International Conference on Distributed Computing Systems Workshops*, pages 331–335, 2002.

[8] Andras Varga and Rudolf Hornig. An Overview of the OMNET++ Simulation Environment. In *Simutools '08: 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems and Workshops*, pages 1–10, 2008.

A Supercomputing API for the Grid*

Károly Bósa† and Wolfgang Schreiner†

1. Introduction

No application can execute efficiently on the grid that is not aware of the fact that it runs in an heterogeneous network environment with heterogeneous nodes. We report on an ongoing work whose goal is to develop a distributed software framework and an API for grid computing which shall empower applications to perform scheduling decisions on their own and utilizing the information about the grid environment in order to adapt their algorithmic structure to the particular situation. Since our solution hides low-level grid-related execution details from the application by providing an abstract execution model, it is able to eliminate some algorithmic challenges of nowadays grid programming.

Regard an example where a user intends to execute a tree-like multilevel parallel application on the grid. She specifies in advance that the given application should consist of 20 processes organized into a 3-levels tree structure. On the lowest level leaves belonging to the same parent process should form groups such that each group contains at least 5 processes scheduled to the same local network environment. For this specification, our software framework is able to determinate a (nearly) optimal distribution of processes on the momentarily available grid resources and to start the processes according to this distribution. Furthermore, our API is able to apply at runtime a corresponding mapping between the predefined roles of processes in the specified hierarchy (global manager, local manager and workers) and the allocated pool of grid nodes such that it minimizes the execution time.

2. Architecture

In our approach, the user assigns to each given parallel program a pre-defined *schema* that specifies a preferred communication pattern of the program in heterogeneous network environments. In our system the following kinds of communication schemas are currently employed [1]: the schema *singleton* specifies a number of processes which should be scheduled to the same local network environment; the schema *groups* specifies the number of processes and either the accurate size of the *local groups* (the number of processes in the same local network environment) or a minimum size for the local groups and some restriction for the number of the local groups; the schema *graph* is similar, but it additionally defines edges/links between the local groups such that they describe a communication pattern; the schema *tree* specifies a tree-like multilevel parallelism with the given number of processes and the given number of tree levels, such that the sizes of the local groups located on the lowest level (the level of leaves) are not determined but some restrictions are given for it; last but not least the schema *ring* is similar to the schema graph, but the local groups always compose a ring.

*The work described in this paper is supported by the Austrian Grid Project, funded by the Austrian BMBWK (Federal Ministry for Education, Science and Culture) under contract GZ BMWF-10.220/0002-II/10/2007.

†Research Institute for Symbolic Computation (RISC), Johannes Kepler University,
email: Karoly.Bosa@risc.uni-linz.ac.at, Wolfgang.Schreiner@risc.uni-linz.ac.at

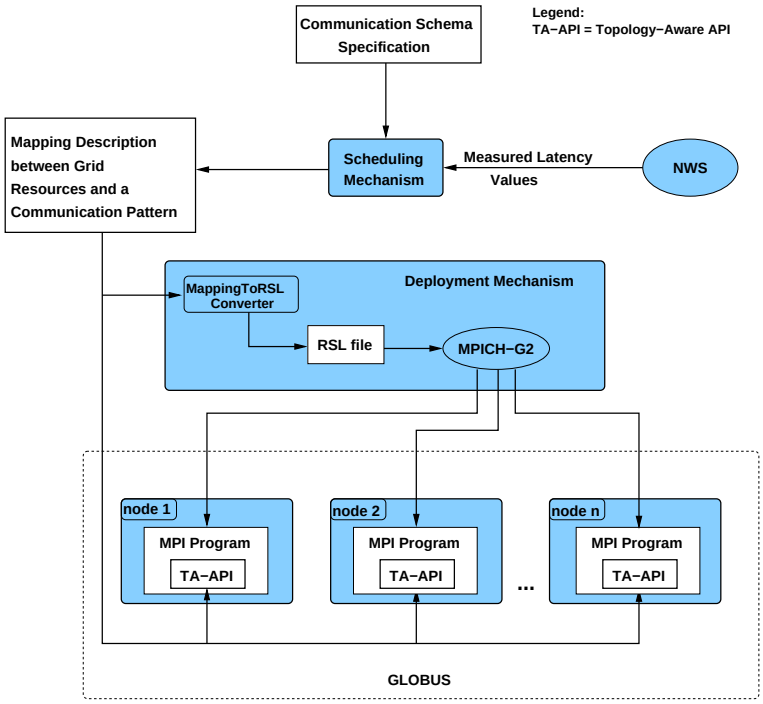


Figure 1. Overview on the Software Framework

We have already designed and implemented the first prototype version [1, 2] of our software framework which is based on the pre-Web Service architecture of the Globus Toolkit and MPICH-G2 [3]. Our solution consists of three major components (see Figure 1):

Scheduling Mechanism depends on the *Network Weather Service (NWS)* [6], which became a de facto standard in the grid community as a performance prediction tool. Since the NWS provides all necessary information concerning the utilizable grid resources, namely the list of the available hosts, a forecast for the available CPU fractions on these hosts and a forecast for the latency values in milliseconds are predicted for each pair of hosts, the user needs therefore not know any detail of the grid architecture. Of course, in addition to these performance characteristics the scheduling algorithm needs a preferred communication pattern of a particular application the user must specify in an XML format.

Before each execution of a parallel program on the grid, the Scheduling Mechanism adapts and maps a preferred communication pattern of the program to the available grid resources such that it heuristically minimizes the assessed execution time. It works roughly as follows:

1. First we classify all the links between each pair of hosts according to the order of magnitude of latencies. For the generated classes we assign an ascending sequence of integer numbers (*latency levels*). To the class which comprises the fastest links we assign the level 1, to the next one we assign the level 2 and so forth.
2. We compose some not necessary disjoint clusters (let us call them *latency clusters*) from all the given hosts such that the latency levels of the links between any two member hosts of such a cluster cannot exceed a certain value (some of these latency clusters may comprise some others with less maximum latency level). Furthermore each host itself is regarded as a latency cluster with the latency level 0. The generated latency clusters are stored in a list which is sorted according to the max. latency levels in an ascending order.

3. We generate all those *partitioning* of processes (in which processes are organized into various local groups) which fulfil the given preferred communication pattern of a program.
4. Finally we map the generated process partitions to some latency clusters according to some heuristic (which helps to avoid the combinatorial explosion of possibilities) and find a reasonably efficient scheduling for the program. In the comparisons of the mappings the algorithm takes into consideration the following characteristics: the maximum latency level within the local groups, the absolute maximum latency level in the entire mapping, the number of the local groups and the average latency value among the local groups.

The output of the algorithm is an XML-based *mapping file* (describing a mapping between the grid resources and the given communication pattern).

Deployment Mechanism is based on the starting mechanism of the grid-enabled MPI implementation MPICH-G2 [3]. It distributes the mapping file on the corresponding grid nodes and starts the processes of the given program on the grid according to the mapping file.

Topology-Aware API is an addition to the MPICH programming library. Its purpose is to query mapping files and inform parallel programs how their processes are assigned to some physical grid resources and which are the designated roles for these processes, such as: in which local group a particular process is involved; which are the characteristics of local groups, of graphs (e.g.: neighborhoods of a group, distance of two groups, etc.), of trees (e.g.: depth of a tree, parent and children of a process, etc.) or of rings.

For representing the versatility of our API, we have developed a simple distributed example application [2] which can be used to establish different kinds of the tree-like multilevel parallelism on the grid according to a mapping file.

Our implemented supercomputing API and the Deployment Mechanism have already been tested successfully on the grid sites `altix1.jku.austriangrid.at` (Altix 350) and `lilli.edvz.uni-linz.ac.at` (Altix 4700). In the final version of the paper, we are going to report on the full functionality of our completely implemented software framework.

3. Related Work and Conclusions

Obtaining high-performance on the grid requires a balance of computation and communication among all involved resources. Currently this can be done by manually managing computations, communications and data locality using message-passing (e.g.: MPI) or remote procedure call (e.g.: GridRPC). Although GridRPC [4] may become an OGF standard as a parallel programming interface for the grid, but it is still based only on the Client-Server model (as any other remote procedure call APIs) and lacks the versatility and power of message-passing based APIs.

While MPI addresses some of the challenges in high-performance grid computing, it was originally designed only for clusters or other homogeneous network environments. A parallel programming environment evolved for the grid must be *topology-aware* in that sense that it must be aware of and exploit the characteristic of an available physical network architecture. Typical topology-aware programming tools are e.g. *MPICH-G2* [3] and *MPICH-VMI* [5]. Both of them are grid-enabled MPI implementations based on the MPICH library. MPICH-G2 uses some grid services provided by the Globus Toolkit pre-Web Service architecture. MPICH-VMI utilizes the middleware communication layer *Virtual Machine Interface (VMI)*.

Summarizing their achievements, we can say that existing topology-aware programming tools make available the given topology information on the level of their programming API and they optimize (only) the collective communication operations (e.g.: broadcast) with the help of the topology information such that they minimize the usage of the slow communication channels. But they are still not able to adapt the point-to-point communication pattern of a parallel programs to network topologies such that they achieve a nearly optimal execution time on the grid. Compared to these existing topology-aware programming tools, the major advantages of our solution are the following:

- It takes into consideration the point-to-point pattern of a MPI parallel program and tries to fit it to a heterogeneous grid network architecture,
- It preserves the achievements of the already existing topology-aware programming tools. This means the topology-aware collective operations of MPICH-G2 are still available, since MPICH-G2 serves as a basis for our software framework.
- Our system eliminates the algorithmic challenges of the high-performance programming on the dynamic and heterogeneous grid environments. Programmers need to deal only with the particular problems which they are going to solve (like in a homogeneous cluster environments).
- The distribution of the processes is always conformed to the loading of the network resources.

In the final version of the paper, we intend to present some comparative benchmarks between the pure MPICH-G2 framework and our topology-aware solution.

References

[1] Karoly Bosa and Wolfgang Schreiner. Initial Design of a Distributed Supercomputing API for the Grid. Austrian Grid Deliverable AG-D4-2-2008_1, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, September 2008.

[2] Karoly Bosa and Wolfgang Schreiner. A Prototype Implementation of a Distributed Supercomputing API for the Grid. Austrian Grid Deliverable AG-D4-1-2009_1, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, March 2009.

[3] N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing (JPDC)*, 63(5):551–563, May 2003.

[4] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and Casanova. A GridRPC Model and API for End-User Applications. GridRPC Working Group of Global Grid Forum, June 2007.

[5] A. Pant and H. Jafri. Communicating Efficiently on Cluster Based Grids with MPICH-VMI. In *CLUSTER '04: Proceedings of the 2004 IEEE International Conference on Cluster Computing*, pages 23–33, Washington, DC, USA, 2004. IEEE Computer Society.

[6] Rich Wolski, Neil T. Spring, and Jim Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.

Analysing a petabyte-scale distributed data management system

Mario Lassnig*†, Vincent Garonne†, Angelos Molfetas† and Miguel Branco‡

Abstract. *Data-intensive computing is subject to high variances in its infrastructures. Current large-scale environments employ static data access policies to cope with this uncertain system behaviour. In order to deal with this automatically and adaptively we need to understand system workload in a more detailed way. We present a scalable analysis method that is able to capture system behaviour in highly dynamic environments up to millions of events per day. We apply our method to DQ2, a petabyte-scale distributed data management system. We highlight the impact of data movement to job execution time and the dominant influence of mass-storage systems. We give examples on workload on the global scale and motivate the need for more detailed data management models.*

1. Introduction and related work

The high-energy physics experiment *ATLAS* presents a data-intensive environment on an unprecedented scale [1]. The middleware *DQ2* is responsible for the experiment's distributed data management and already scales to tens of petabytes of managed data [3]. DQ2 is written in Python and built atop the *EGEE gLite*, *NorduGrid ARC* and *Open Science Grid* (*OSG*) architectures [2, 4, 8]. DQ2 combines them into a multi-middleware *data-grid* with some *cloud computing* properties [15], like run-time configuration of sites, inter-middleware transfers and virtual resource provisioning. DQ2 therefore provides a single point of entry to all experiment data using a common interface and quality of service over multiple heterogeneous infrastructures. Figure 1 shows a high-level overview of the architecture: the grid middleware layer at the foundation, the common modular framework in use by all components, central bookkeeping of all data, distributed site services to manage scheduled data movement, client interfaces to connect external applications as well as interactive clients for users.

The experiment's ways of accessing data are summarised in its computing model [7]. They can be generalised into three groups: *creation*, i.e. data-acquisition from the experiment and production of simulated data; *scheduled movement*, i.e. export and transfer of data between sites; and *ad hoc access*, i.e. analysis of data and automated data processing with physics algorithms. Creation and scheduled movement of data are well understood by the experiment and in literature but ad hoc access to data is not [3]. Since users must be able to request large samples of data synchronously and immediately, in the order of tens of gigabytes, the system needs to support unscheduled data access subject to constraints of already scheduled parallel activities. This is further complicated by high

*Distributed and Parallel Systems, University of Innsbruck, AT-6020 Innsbruck, email: mario@dps.uibk.ac.at
†European Organisation for Nuclear Research (CERN), CH-1211 Geneva, email: {firstname.lastname}@cern.ch
‡IAM/ECS, University of Southampton, SO17 1BJ United Kingdom, email: msbranco@gmail.com

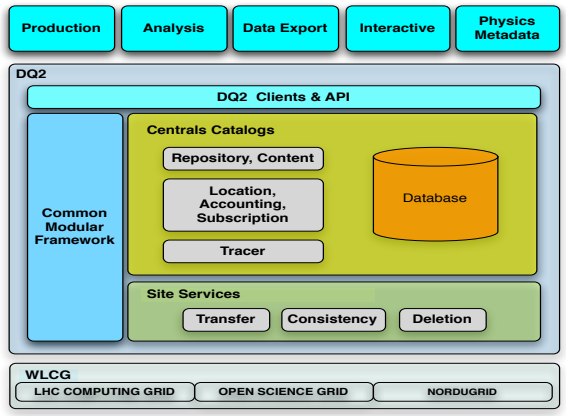


Figure 1. Overview of DQ2 architecture

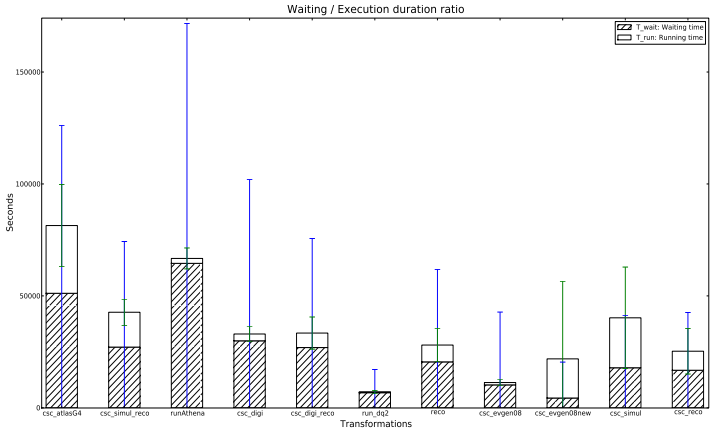


Figure 2. Job waiting/running time ratio

variances in dependent resource's performances. To approach this problem, we have developed a new analysis method to trace system behaviour. The objective of the method is to scale to our experiment's requirements with millions of file accesses and transfers daily and to support full access provenance for this. Furthermore, we have to use multiple grid middleware stacks, so the second objective was to be as non-intrusive as possible because the existing infrastructures could not be modified.

We categorise three tracks of directly related work: *workload*, to model resource behaviour; *availability*, to model resource failures; *rare event prediction*, to detect singular anomalous behaviour. Due to space constraints we can only give exemplary citations though. Recent workload analysis was done focusing on modelling and predicting dynamic resource availability and employing models with diverse capabilities, like adaptive workloads, multi-state resources or congestion control [6, 10, 11]. All proposed models were based on homogeneous environments though, neglecting the complexity introduced through heterogeneous systems. Recent studies in failure behaviour examined the impact of disk faults to system availability [14]. There is no notion of degrading systems though and the possibility of unscheduled access to resources was never considered. The work most similar in methodology to ours was done by Gao et al. [5], who tracked large-scale data movements to find software bugs. In rare event prediction, the work that is most interesting to us adaptively identifies sudden changes in system behaviour over time, which is what we observe in DQ2 [9, 13].

2. Model & Analysis

One of the hardest problems in a production environment is that complete information about system behaviour is very difficult to attain. Fortunately, due to the scale of the system, we can rely on the *law of large numbers* and infer this information probabilistically. This is the core of our analysis method. We implemented it as a distributed tracer that reports all local and distributed file movement *events* since 2008-05-14. The implementation is purely in the DQ2 layer and thus middleware agnostic. The current event rate is 17 ± 2Hz, i.e. 1.5 million events per day with negligible overheads below 0.025. At full scale operation we expect 120 Hertz. We instrument the following properties: start time, end time, time spent in global and local namespace lookups, time waiting for mass-storage systems, time spent on the network, time spent validating files, and tracing overhead. This allows us to build a complete execution profile and is directly representative of overall system behaviour. Furthermore, we trace additional data management information, like dependent events, version information, involved sites, file information, transfer protocols used, clients used, client exit conditions and anonymised user

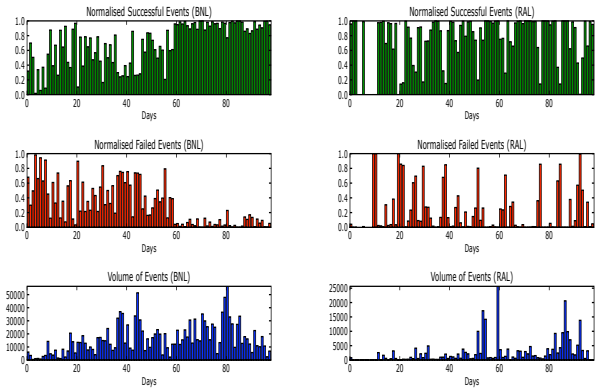


Figure 3. Transfers from two sites (BNL, RAL)

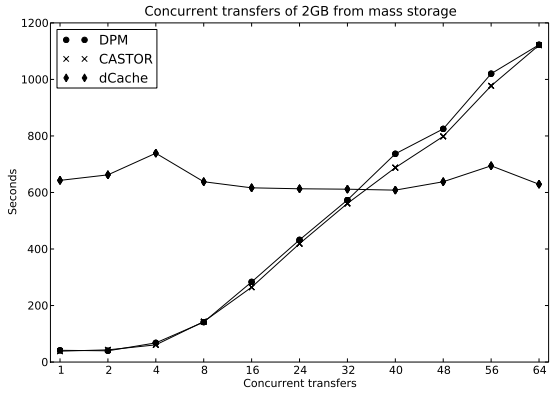


Figure 4. Storage behaviour on concurrent transfers

information. Figure 3 shows normalised event traces of interactive DQ2 usage of two sites over 90 days with one-day aggregates. The sudden increase in site performance of site *BNL* at day 60 is clearly visible and independent of the volume of incoming requests. On the other hand, we observe that site *RAL* exhibits bi-weekly bursts of unsuccessful transfers. We are thus able to present a probabilistic global view of the system that is still very detailed by examining large numbers of file events.

We also observe that site performance is directly dependent on the number of concurrent transfers to contested resources. These resources do not fail as such but degrade in performance until they reach a state where performance is non-satisfactory. Even though they do not fail, they become unavailable for practical purposes. Figure 4 shows the effect of increasing the number of concurrent requests to transfer times to a specific *mass-storage* system. We selected three mass-storage systems with negligible parallel activity and enacted transfers of 2GB files. Each transfer was handled by a dedicated client machine with spare bandwidth. The time spent is thus dependent on the mass-storage system only. We repeat each experiment 64 times and present the median. CASTOR and DPM seem to use the same throttling mechanisms and thus exhibit the same linear degradation. dCache on the other hand shows a mostly uniform performance independently of the number of requests. This data was collected by the tracer the same way and shows that service information can be extracted as well.

3. Motivation for future work and conclusion

Data management is still a neglected topic in grid research, therefore we think it is appropriate to motivate the need for more detailed studies: ATLAS uses pull-based job scheduling using *pilots* that execute jobs on a site's cluster once the input data for a job is available [12], thus sending the jobs to the data. We uniformly sampled one fourth of 14 million jobs from two months, with 82.63% successful jobs. Only 5.41% of failures were due to data management, equal to 0.94% of failures overall. ATLAS uses certain *job transformations*, i.e. workflows, and we grouped our observations on these transformations. Figure 2 shows the ratios between T_{wait}, the mean waiting time from creation of the job until it executes, and T_{run}, the mean time spent in computation. The longer blue bars show the standard deviation for T_{wait}, the shorter green bars for T_{run}. The dominance of T_{wait} is due to local resource management and data movement. We can get the ratio of local cluster scheduling from transformations that do not need input data, like *run_dq2* or the *csc_evgen*. This results in $28.37 \pm 13.71\%$ of overall time spent for local resource management, leaving more than two thirds of waiting time to data management. The negligible number of data management faults suggests that

resources have reached maximum capacity and thus other models of data management need to be explored to counter T_{wait}. Analysis methods like ours can be used to help creating such models.

Negligible data management faults and domination of T_{wait} to run-time thus make a strong case to focus on improving data access. A global view of resource and service performance is needed to accomplish this and our tracing infrastructure provides a non-intrusive high-performance way for gathering such run-time information. We show how this method can be used to create statistics of resources and services and showcase the influence of concurrent transfers to resource availability. We observe that resources can degrade to an unusable state without failing. Our future work focuses on modelling and simulation of large-scale data-intensive systems subject to incomplete information.

References

[1] ATLAS Collaboration, "ATLAS Technical Proposal", *CERN*, http://atlas.web.cern.ch/, 1994

[2] Avery et al., "Open Science Grid: Building and Sustaining General Cyberinfrastructure Using a Collaborative Approach", *First Monday*, Vol. 12(6), University of Illinois at Chicago, 2007

[3] Branco et al., "Managing very-large distributed datasets", *Lecture Notes In Computer Science*, Vol. 5331(08), Springer, 2008, pp. 775

[4] Ellert et al., "Advanced Resource Connector middleware for lightweight computational Grids", *Future Generation Computer Systems*, Vol. 23, 2007

[5] Gao et al., "DMTracker: Finding Bugs in Large-Scale Parallel Programs by Detecting Anomaly in Data Movements", Proc. of *Supercomputing 2007*, ACM, 2007, pp. 15

[6] Iosup et al., "On the dynamic resource availability in grids", *8th IEEE/ACM International Conference on Grid Computing*, IEEE, 2007, p. 26-33

[7] Jones et al., "The ATLAS Computing Model", *CERN*, http://cdsweb.cern.ch, Rec. 811058, 2005

[8] Laure et al., "Middleware for the next generation Grid infrastructure", Proc. of *Computing in High Energy Physics and Nuclear Physics*, Interlaken, Switzerland, 2004, pp. 826

[9] Knorn et al., "Adaptive Kalman Filtering for Anomaly Detection in Software Appliances", *INFOCOM Workshops*, IEEE, 2008, p. 1-6

[10] Li, "Workload dynamics on clusters and grids", *J. of Supercomp.*, Vol. 47:1-20, Springer, 2008

[11] Nadeem et al., "Characterizing, Modeling and Predicting Dynamic Resource Availability in a Large Scale Multi-Purpose Grid", *8th IEEE Int. Symp. on CCGRID*, IEEE, 2008, p. 348-357

[12] Nilsson et al., "The PanDA System in the ATLAS Experiment", Proc. of *13th Advanced Computing and Analysis Techniques in Physics Research*, SISSA/POS Publishing, 2008, pp. 27

[13] Sahoo et al., "Critical Event Prediction for Proactive Management in Large-scale Computer Clusters", Proc. of *9th Int. Conf. on SIGKDD*, ACM, 2003, p. 426-435

[14] Schroeder et al., "Understanding Failures in Petascale Computers", *Journal of Physics: Conference Series*, Vol. 78(7):012022, IOP Publishing Ltd, 2007

[15] Weiss, "Computing in the Clouds", *netWorker*, ACM, Vol. 11(4):16-25, 2007

Research on Security Requirements for Grid Environments[1]

Grid Computing technology promulgates in both academic and business environments amplifying its importance in business world. In both cases security is crucial and will decide over success or failure of this technology. Authentication and authorization are two important security aspects whereas several methods exist. Regarding Data Grids, where information is shared within a so called virtual organization (VO), tighter control must be kept over information enabling all-embracing audit trails to comply with laws and regulations. However log files, which are required for audit trails, pose several problems regarding format, evaluation and some other factors. This paper researches state-of-the art technologies, discusses unsolved and challenging problems of Grid security technology and provides possible solutions.

Security Evaluation of Authentication and Authorization Mechanisms

Authentication and Authorization are the preliminary corner stone of security, identifying and regulating users' access to components shared within data Grid environments. Those two processes are crucial security components, as user information and access decisions are transmitted over insecure networks and users belong to different security domains. Thus secure mechanisms are needed to protect shared data, identifying users reliably to prevent unauthorized access or malicious activities. Due to the importance of those security measurements, several authentication and authorization mechanisms have been developed which are specialized for Grid computing environments.

However availability of several different products introduces the problem of choosing the right product for different environments, fulfilling the individual security needs. This can be a very challenging task, as solutions cannot be compared directly. Hence, to solve this problem, a question catalog has been designed to evaluate *security* of different *authentication/authorization* mechanisms, making them comparable more easily. We chose six (CAS [6], VOMS [1], EALS [3], Akenti [7], PERMIS [2], Shibboleth [9]) of the most popular methods and evaluated them with the aforementioned approach. An excerpt of the results is illustrated in Table 1.

[1] David Huemer, A Min Tjoa, Marco Descher; Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstraße 9-11, {dhuemer|atjoa|mdescher}@ifs.tuwien.ac.at, http://www.ifs.tuwien.ac.at/

Security Aspects	Ranking	Evaluation aspects	CAS	Comments CAS	VOMS	Comments VOMS
Type of authentication	3	UN/PW	1	UN/PW and certificate are used	1	UN/PW and certificate are used
	2	Cert				
	1	biometric/combination				
Strength of authentication	3	1-factor	2	2-factor authentication is used (see type of authentication)	2	2-factor authentication is used (see type of authentication)
	2	2-factor				
	1	Multi-factor (>2)				
Validity of credentials	2	unlimited	1	validity time is intersection of validity times of all certs in the cert chain	1	usually 12 h
	2	long				
	1	short				
Single point of exploit/failure (SPOE/F)	2	yes	2	CAS server is SPOE/F	1	user may contact several VOMS, compromising VOMS not enough, because authorization data must be inserted in user proxy cert
	1	no				
Fine grained access control possible	2	no	1	User has ultimate access decision, restricted proxy (extension to X.509)	1	resource owner has utlimate decision
	1	yes				
Protection transport level	2	no	1	TLS used	1	SSL used
	1	yes				
Revocation time	3	until Grid admin reacts	2	certs usually expire within hours --> no explicit revocation	2	no explicit revocation, usually 12 h
	2	exp. validation period				
	1	immediately				
Complexity of system (number of systems, logical complexity, complexity for users, …)	3	high	2	users have to know how to use certs, but auth/authz system only consists of resources and CAS-server	2	handling with certs, differerent VOMS, leaves interpretation of Attribute Certificates to other components (i.e. local sites) --> only PIP, PEP at resource
	2	medium				
	1	low				
SSO support	2	yes	2	GSI supports SSO	2	GSI supports SSO
	1	no				
Policy enforcement	2	no	1	enables policy enforcement on VO basis	1	X.509 supports policy conditions for certs, CP (Certificate Policy) and CPS (Certification Practice Statement)
	1	yes				
Mutual authentication	2	no	1	yes, if CA issuing X.509 cert is trusted by all communication partners	1	yes, if CA issuing X.509 cert is trusted by all communication partners
	1	yes				
Auditing functionality	2	no	2	not per default	1	keeps deletions/modifications in archive table, also deletedBy
	1	yes				

Table 1: **Evaluation results of two of the six evaluated authentication/authorization methods**

Audit Trails

Another important security concern when sharing highly sensitive data, such as medical information within Grids, is *traceability*. Several standards and regulations exist, demanding to comply to regulations to ensure sensitive data are well protected and data usage is comprehensible. HIPAA (Health Insurance Portability and Accountability Act) [8], for example, is such a standard requesting, amongst others, the following:

- Procedures should clearly identify persons who will have access to electronically protected health information (EPHI). EPHI must only be accessible by persons who have a need for it to complete their job function.

- The procedures must address all access decisions (authorization, establishment, modification, termination).

- Each participating party is responsible for ensuring that data within its systems has not been tampered with in an unauthorized manner.

- Mutual authentication should be applied between the communicating parties. Strong encryption realized by tokens and two factor authentication should be deployed for this purpose.

Those requirements can only be met by deploying audit trails realized by systematic logging procedures. Compared to authentication/authorization those requirements cannot be met easily because no framework or other technology exists, enabling automatic evaluation of log files containing huge amount of data utilizing different log formats.

To implement reliable logging it has to be guaranteed, that log files are *i)* created *ii)* securely stored and *iii)* evaluated (automatically). Although those steps sound trivial it is a very challenging task. For example it must be guaranteed that no side-access to data is possible, addressing *i)*. Thus we developed the concept of a deployable Secure Virtual Machine [4]. The architecture is depicted in Fig. 1.

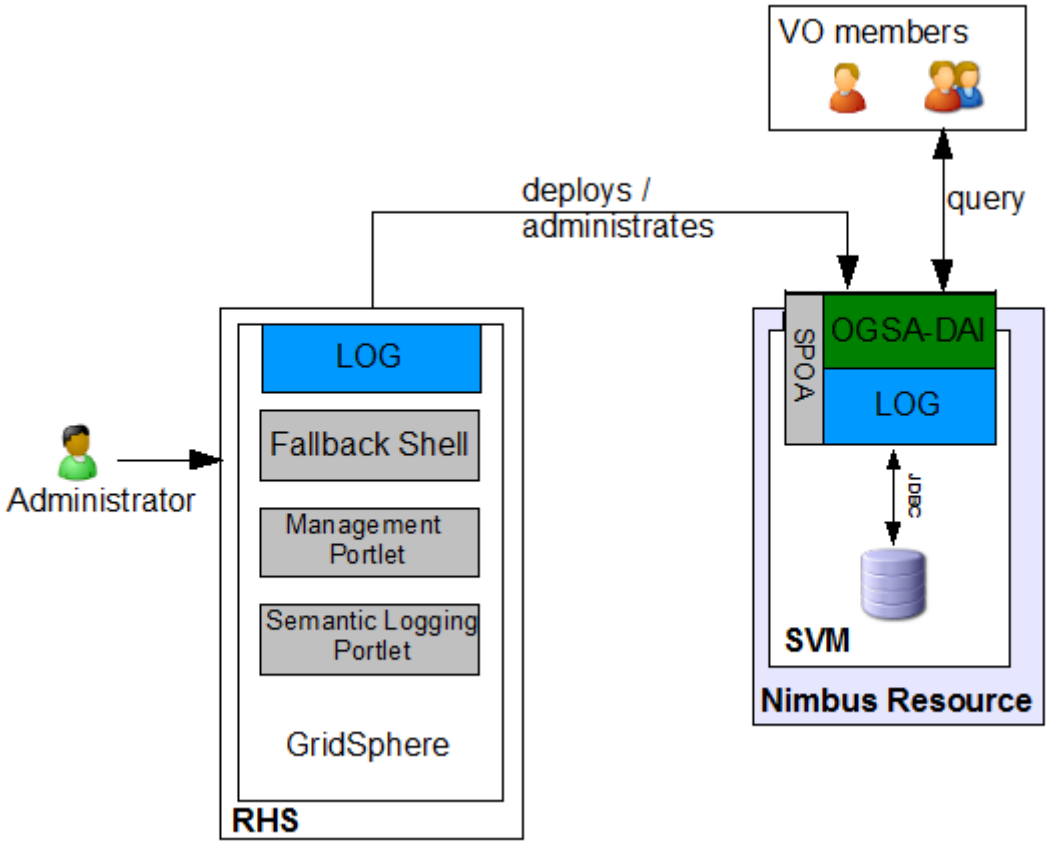


Fig. 1: **A Secure Virtual Machine (SVM) provides data within a Grid environment, whereas the Single Point of Access (SPOA) is the OGSA-DAI interface. Each interaction with the machine is recorded into a LOG file guaranteeing comprehensibility.**

The concept consists of a portal and a secure virtual machine (SVM), whereas administrative tasks can be conducted through the portal and the only access to the SVM is via a modified OGSA-DAI interface, also serving requests from VO members. Logging is enforced at the OGSA-DAI interface, which is the single point of access (SPOA). Thus it can be guaranteed, that every interaction gets recorded into a LOG file. Additionally administrators may request a fallback shell initiated through the portal to conduct administrative tasks. Every command typed will be saved at portal and SVM level making it possible to detect malicious behavior (e.g. trying to modify log files) even if conducted by an administrator. Details about the fallback shell and the whole infrastructure are described in [5].

Conclusion

In this extended abstract security requirements for Grid environments and some challenges are introduced. Results of a newly developed evaluation method for authentication and authorization methods are presented and some concepts for a reliable Grid logging architecture are described. The first one solves the problem of comparability of different authentication methods, whereas the second one lays the corner stone for solving problems with compliance requirements arrogated by laws and regulations.

References

[1] Alfieri R., Cecchini R., Ciaschini V., dellAgnello L., . Frohner, Gianoli A., Lrentey K., and Spataro F., “VOMS, an authorization system for virtual organizations,” in Grid Computing, ser. Lecture Notes in Computer Science, vol. 2970/2004. Springer Berlin / Heidelberg, 2004, pp. 33–40.

[2] Chadwick D. W. and Otenko A., “The permis x.509 role based privilege management infrastructure,” Future Gener. Comput. Syst., vol. 19, no. 2, pp. 277–289, 2003.

[3] Chakrabarti, Anirban, Grid Computing Security, ISBN: 978-3-540-44492-3, XIV, 332 p. 87 illus., Hardcover, 2007

[4] Descher Marco, Masser Philip, Feilhauer Thomas, Tjoa A Min and Huemer David, Retaining Data Control to the Client in Infrastructure Clouds, 2009 International Conference on Availability, Reliability and Security, IEEE Computer Society, 9-16, In Press, Fukuoka, Japan

[5] Huemer David, Tjoa A Min, Descher Marco, Feilhauer Thomas and Masser Philip, Towards a Side Access Free Data Grid Resource by Means of Infrastructure Clouds, Second International Workshop on Simulation and Modelling in Emergent Computational Systems (SMECS-2009), IEEE Computer Society, Sept. 2009, Vienna, Austria

[6] Pearlman L., Welch V., Foster I., Kesselman C., and Tuecke S., “A community authorization service for group collaboration,” in POLICY ’02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY’02). Washington, DC, USA: IEEE Computer Society, 2002, p. 50.

[7] Thompson M. R., Essiari A., and Mudumbai S., “Certificate-based authorization policy in a pki environment,” ACM Trans. Inf. Syst. Secur., vol. 6, no. 4, pp. 566–588, 2003.

[8] U. D. of Health and H. Services, Health insurance portability and accountability act, February 2009, http://www.hhs.gov/ocr/privacy/index.html.

[9] Zhang, N.; Chin, J.; Rector, A.; Goble, C.; Li, Y., Towards an authentication middleware to support ubiquitous Web access, Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International , vol.2, no., pp. 36-38 vol.2, 28-30 Sept. 2004

A RESTful Repository To Store Services For Simple Workflows

Ralph Vigne[*],
Jürgen Mangler[†], Erich Schikuta[‡] and Christoph Witzany[§]

Abstract. *When composing workflows for mobile devices many problems arise: How to find the services? How to describe the services so that every user can compose its own workflow without technical knowledge? How to handle similar services with different interfaces? How to deal with performance issues? Most of these problems can be solved by using an appropriate repository for services and by describing the services with existing interface description languages and annotations like approaches based on OWL-WS [1]. But these approaches are neither particularly easy to support, as they invoke the creation of excessive ontologies that hold information about the services, nor are they based on the RESTful paradigm that for good reasons dominates current cloud applications. In our paper we concentrate on creating a restful repository that stores services in a hierarchical manner. The repository is able to store restful services, references to common properties of the services as well as information how to interact with these services.*

1. Introduction

As grid computing currently is moving back to its origins, the high performance computing domain, the new buzzword that arises is "cloud computing". Cloud computing is dominated by simple (if any) middleware and simple interfaces. Unlike Grid the term cloud is not yet misused by companies for marketing purposes, maybe also because its definition is much vaguer [2]: according to Vaquero et al. the cloud is a set of easily accessible virtual resources, the cloud is transparent and the cloud has a business model attached to it that is assured by Service Level Agreements (SLAs).

Apart from this definition the cloud is also successful because of the newly rediscovered focus on simplicity [3]. Many of todays existing successful cloud applications have simple RESTful [4] interfaces, that impose no dependencies on the client except the knowledge of HTTP and XML. This quickly led to open source initiatives like e.g. Eucalyptus [5] that imitate the RESTful interface of Amazon EC2.

Using virtual resources is tightly coupled with workflows. Current workflow systems often based on WS-BPEL heavily rely on WS-* standards and are by no means lightweight. Like Rosenberg et al. [6], we think its about time to embrace the new possibilities offered by RESTful architectures and

[*]Faculty of Computer Science, University of Vienna, Austria, email: ralph.vigne@univie.ac.at
[†]Faculty of Computer Science, University of Vienna, Austria, email: juergen.mangler@univie.ac.at
[‡]Faculty of Computer Science, University of Vienna, Austria, email: erich.schikuta@univie.ac.at
[§]Faculty of Computer Science, University of Vienna, Austria, email: christoph.witzany@univie.ac.at

design for simplicity and scalability.

Designing such an architecture to include RESTful services is also promising, as most businesses today provide websites with forms to order, reserve and/or cancel products and services. All these websites are potential services in a restful architecture as unlike SOAP based services RESTful services are based on the standard building blocks of the web.

Our vision is to create an iPhone application based on this RESTful architecture, that allows the user to simply specify a workflow from predefined freely available business cases, and than save the resulting workflow to his/her device. The user can then call the workflow over and over again, RESTful services fitting the business cases will be selected and used dynamically at runtime. For our goal to create this simple workflow application for mobile devices, we specified the following design rationale:

- A repository to store arbitrary services is needed.
- Service entries have to be stored in hierarchical manner, to support scalability and performance (see Section 4.).
- A service is described by properties. Static properties are immutable, dynamic properties can vary for each request or in time.
- Properties are valid for a particular branch of a hierarchy of the repository. Therefore each service in a branch is suitable to substitute every other service.
- Each property is defined by a schema in order to be checked for valid results when queried.
- Each property has information attached for the client to be able to generate a user interface.
- There has to be a unified way to access services. In order to realize the unified access we need a message transformation mechanism.

To describe the properties we selected RelaxNG [7], as it is lightweight in comparison to XML-Schema. A second advantage we gain through the use of RelaxNG is, that the above mentioned information to generate UI's can be annotated to RelaxNG with ease, as RelaxNG supports arbitrary annotations in comparison to XML-Schema.

In this paper we introduce a scalable, lightweight repository that stores arbitrary services, properties for these services and provides for a way to access these services in a uniform way through a message transformation mechanism. We will also describe how to query the repository and how the structure of the repository affects the performance and scalability of the overall system.

The paper is structured as follows: In Section 2. we classify the information stored in the repository and describe the overall structure of the repository. In Section 3. we will further explain how simple schemas are used to describe properties and the overall interaction with the repository. In Section 4. we will deal with performance considerations when designing a repository, in Section 5. we have a look on related work. Section 6. describes our envisioned iPhone client, the concepts behind and the problems still to be solved. Finally we will conclude this paper in section 7. with a short outlook on future research.

2. Information Stored in the Repository

In order to make the services accessible to customers we want to provide a registry similar to UDDI [8]. As we deal with restful web-services it is not suitable to use UDDI itself as it builds on the heavy-weight SOAP/WSDL stack. So we decided to create a much more lightweight repository, based on standards used in the restful world, but that shares concepts with UDDI:

- Yellow Information: Information how the vendors are grouped (groups and subgroups).
- White Information: Information about the vendor (name, public key).
- Green Information: Technical information like message structure and properties schema.

Instead of access through a traditional web-service interface, the repository will be accessible as a restful web-service, exposing groups and subgroups as directories, and delivering list of vendors as ATOM feeds [9]. The benefit of this approach is, that the repository can be used through a browser and clients can be easily realized, as implementations for HTTP and ATOM should be readily available for many platforms and systems.

In subsequent sections we will explain the concepts of our repository by using the following example: A user wants to watch the movie "Star Trek" in a nearby arthouse cinema. The current position of the user is supported in the form of longitude / latitude coordinates by the phone. The client application should be able to present the user with a cinema fitting the preferences of the user.

2.1. Yellow Information

The repository is organized in groups. Each group consists of a number of subgroups which holds a set of interchangeable services with similar properties. Each group defines four messages which have to be implemented by every service in a particular subgroup of the group. Static properties are stored at group level and will be inherited by the subgroups, like the messages. As mentioned above we implemented the repository restful, which means that every group or subgroup can be accessed directly through an URI. For example:

```
uri:repo/cinemas
uri:repo/cinemas/arthouse
uri:repo/cinemas/multiplex
```

If a client wants to receive a list of subgroups from the group "cinemas", it asks for it at *uri:repo/-cinemas*. The answer will be the list of all subgroups. With this information the client can figure out the URI of the requested subgroup.

2.2. White Information

A call to *uri:repo/cinemas/arthouse* then receives all arthouse cinemas as an ATOM feed (according to the example given above), e.g:

LISTING I. ATOM FEED: ALL SERVICES

```
<?xml version="1.0"?>
```

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:res="uri:rescue">
  <title>List of Resources</title>
  <updated>2009-05-28T15:56:39Z</updated>
  <generator uri="http://www.pri.univie.ac.at/rescue">Rescue</generator>
  <id>uri:repo/cinemas/arthouse</id>
  <link href="uri:repo/cinemas/arthouse" rel="self" type="application/atom+xml"/>
  <res:schema>
    <res:properties>uri:repo/cinemas?properties.schema</res:properties>
    <res:queryInput>uri:repo/cinemas?queryInput.schema</res:queryInput>
    <res:queryOutput>uri:repo/cinemas?queryOutput.schema</res:queryOutput>
    <res:invokeInput>uri:repo/cinemas?invokeInput.schema</res:invokeInput>
    <res:invokeOutput>uri:repo/cinemas?invokeOutput.schema</res:invokeOutput>
  </res:schema>

  <entry xml:lang="en">
    <id>uri:repo/cinemas/arthouse/urania</id>
    <link>http://urania.at/restful</link>
    <updated>2009-05-28T15:13:17Z</updated>
    <category term="cinemas"/>
    <category term="arthouse"/>
    <res:properties>http://urania.at/restful/properties.xml</res:properties>
  </entry>
  ...
</feed>
```

It can easily be seen that every *entry* holds vendor specific information, which can of course be requested for single vendors by querying the URI given by *id* (e.g. uri:repo/cinemas/arthouse/urania).

Using this implementation style makes it very easy for client application developers to use the repository and, in turn makes their applications very flexible. Whenever a new group or subgroup is added to the repository, there is no need to adapt the client application. We use the subgroup-concept to make it possible for users to narrow down their requested services and also to increase clarity.

2.3. Green Information

To understand how the *Green Information* (GI) can be used by clients, we have to define the protocol how a client invokes the repository. A client ...

1. asks for a list of subgroups in a given group, e.g. uri:repo/cinemas.
2. asks for a list of services in a given group, e.g. yielding an ATOM feed (See List. I).
3. collects all properties files of the services received in step 2, e.g. by using *entry/res:properties* from List. I.
   - **GI:** In order to understand the returned properties, the client needs the schema returned by *res:schema/res:properties*.
4. queries a subset of the services received in step 2, filtered and ordered according to the properties received in step 3, for dynamic properties.
   - **GI:** In order to be able to supply the submitted query with parameters, the client has to support the schema provided by *res:schema/res:queryInput*.

- **GI:** In order to be able to understand the result of the query, the client has to support the schema provided by *res:schema/res:queryOutput*.

5. invokes a service which has been selected according to results received from step 4.

   - **GI:** In order to be able to supply the invocation with parameters, the client has to support the schema provided by *res:schema/res:invokeInput*.
   - **GI:** In order to be able to understand the result of the invocation, the client has to support the schema provided by *res:schema/res:invokeOutput*.

When developing a client application there are two ways to use this information:

First, the developers look up the messages they need for their application and implement them directly into the client. This way the elements of the user interface and all the other information needed to use the services of a specific group are known in advance and can be hard-coded into the clients source code.

Second, a client is generic in that it can handle every message only by knowing its definition. The definition of the messages contains also information about the caption of an element. Using this caption, a client's user interface can be made generic and cover every message defined in any group of the repository. Within this definition of the caption for an element, different string to support different languages are allowed.

How to use the information will strongly depend on the purpose of the application.

3. Properties of Stored Services and How They are Described

As outlined in Section 2. the repository provides *Green Information* that we will further detail in this chapter.

3.1. Properties

Many vendors offer similar services. To make it possible for a user or a client application to decide which one to use, every service in the repository is annotated with a number of properties. As shown in listing II, these properties fall into two groups. Group one consists of static properties, their values do not change on a regular basis. The second consists of dynamic properties, their values may differ from call to call. This is similar to techniques found in the WS-Resource Properties standard, defined by OASIS [10]. As described above the properties are unique for each group, their structure (List. II) may be obtained through *res:schema/res:properties*.

3.1.1. Static Properties

Properties in this group are defined directly in the according properties file of the service which will be hosted and maintained by the vendor of the service. They do not change on a regular basis like e.g. longitude and latitude of a cinema or a list of countries that a vendor ships to. With this information, client applications may preselect the services before querying them. The URI of the properties file is stored in the repository (See List. I) and can be received by any client.

3.1.2. Dynamic Properties

These properties may vary from call to call, e.g. like the number of available tickets for a movie or the price of an item at an auction. They are not stored in a static file, but are generated on demand. As described above, the dynamic properties are requested by calling *query*, with parameters according to the *res:schema/res:queryInput*, to narrow down the results and increase the quality of the returned information.

The output of this operation is again defined at group level through *res:schema/res:queryOutput*, which in turn is a subset of the information obtained through *res:schema/res:properties*, namely the *dynamic* part.

This schema driven approach has the following advantages: First, all information about semantic meaning and valid values of the properties is available through the *res:schema/res:properties* schema. Second, even the user interface description for clients is contained within the schema and can therefore be easily maintained and kept in sync.

3.2. The Properties Schema

In the repository, each group defines its own RelaxNG schema [7], consisting of properties and their allowed values. An example for a schema is (according to the example given in Section 2.):

LISTING II. GREEN INFORMATION: EXAMPLE PROPERTIES SCHEMA

```
<properties>
  <static>
    <element name="longlat">
      <ui:caption xml:lang="en">Geo-position (longitude:latitude)</ui:caption>
      <ui:caption xml:lang="de">Geo-Position (Länge:Breite)</ui:caption>
      <data type="string">
        <param name="pattern">\d+\.\d+:\d+\.\d+</param>
      </data>
    </element>
    ...
  </static>

  <dynamic>
    <element name="title">
      <ui:caption xml:lang="en">Title</ui:caption>
      <ui:caption xml:lang="de">Titel</ui:caption>
      <text/>
    </element>
    <element name="numberOfSeats">
      <ui:caption xml:lang="en">Number of available Seats</ui:caption>
      <ui:caption xml:lang="de">Anzahl verfügbare Sitzplätze</ui:caption>
      <data type="positiveInteger"/>
    </element>
    ...
  </dynamic>
</properties>
```

The properties schema for each group is available under the URI *uri:repo/{group}?properties*, with *{group}* being a placeholder for a group name.

The schema is not only used for describing properties and valid values but also holds information about the user interface generated by the client. As mentioned above there are two ways to implement a client applications UI. One is to hard-code the UI for particular services, and the other is to dynamically interpret annotations to the properties.

By providing a human-readable description of each property, it is possible to write client applications in a way that they can generate the UI automatically for any group, even if they are not known at the time of implementation. Figure 2 shows an example UI generated from the *ui:caption* annotations shown in List. II. The schema used for generating the UI is also used to check the input collected through the UI.

3.3. Querying Input/Output Schemas

To cover every possible information a vendor needs or provides with his service, the messages sent or received during query and invocation have to be defined comprehensively. The *Green Information* is based on RelaxNG schemas, which are easy to parse and allows for easy checking of messages. Every client has to be able to check if its implementation of a certain group is valid through checking the messages it generates against the repository provided schemas. For example when querying a service for dynamic properties the schema can look like this:

LISTING III. GREEN INFORMATION: QUERYINPUT

```
<element name="queryInput">
  <element name="title">
    <text/>
  </element>
  <element name="numberOfSeats">
    <data type="positiveInteger"/>
  </element>
</element>
```

A message querying all show-times for the movie "Star Trek" where at least five tickets are available has to look like this:

LISTING IV. QUERYINPUT EXAMPLE

```
<queryInput>
  <title>Star Trek</title>
  <numberOfSeats>5</numberOfSeats>
</queryInput>
```

Each service in a group has to fit the defined schema. Therefore each service needs to transform its messages from or to the definition given in the repository. As shown in Fig. 1 the repository stores schemas for four different messages:

queryInput to specify which dynamic properties are requested.

queryOutput provides the dynamic properties to the client according to queryInput.

invokeInput initiates the actual execution of the service.

invokeOutput delivers the results of the service call initiated by invokeInput.

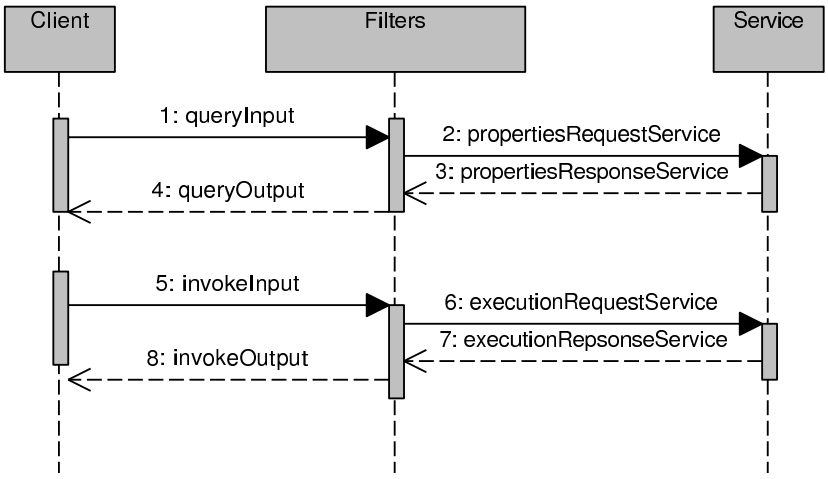


Figure 1. Communication: Client ⇔ Filters ⇔ Service

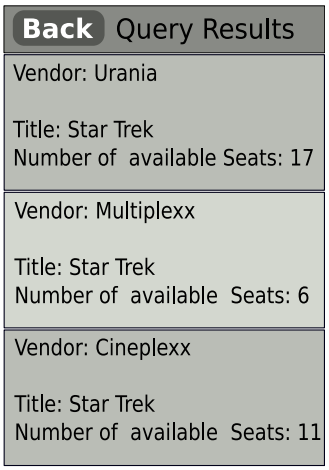


Figure 2. Mockup Client UI From Schema

Reducing the communication between the client and the web-service to those four standardized messages, makes it very easy to implement client applications.

Vendors are intended to keep their current implementations (e.g. forms). Therefore it is necessary to transform the messages purported by the repository to a valid input for the actual service. This will be implemented by providing a message transformation mechanism. The description of this mechanism is beyond the scope of this paper.

4. Scalability and Performance of the Repository

With an increasing number of vendors and client applications, performance and load balancing becomes an important issue.

4.1. Scalability

By providing a RESTful and resource oriented interface for the repository it is very easy to distribute the repository over a number of different servers. Some systems even support automatic load balancing [11] over different web servers. The Provisioning of a uniform interface, like in our approach, allows to make full advantage of this kind of systems. Vinoski pointed out that "[...] there is no question that the uniform interface constraint can allow for more highly scalable systems" [12]. Pautasso et al. conclude their comparison between REST and WS-* as follows: "The main conclusion from this comparison is to use RESTful services for tactical, ad-hoc integration over the Web (a la Mashup) and to prefer WS-* Web services in professional enterprise application integration scenarios with a longer lifespan and advanced QoS requirements" [13]. According to this conclusion REST is the right decision for our purpose, because we integrate services ad-hoc which are provided by many different vendors.

An underlying DBMS will influence the overall performance of the repository, but because of the hierarchical structure of the repository it should be easy to segment the data storage as well.

4.2. Performance

When dealing with a large number of services in the repository, client applications and users need a quick and easy way to access a subset of these services. By providing the data in hierarchical manner (groups and subgroups concept) we decrease the number of services a client requests, which leads to better performance and less traffic. The vertical segmentation also allows for performance increases of concurrent requests. By providing a uniform interface for each group, performance increases because the interface only needs to be requested once per client application instead of per service. This is also in line with the conclusions drawn by Blake et al. [14], most resources when using UDDI repositories are consumed for message parsing and transmission. Blake further finds that regular changes in the repository produce performance decreases, because the repository needs to check if anything has changed since the last request. We avoided this problem by omitting any kind of version control in the repository and by providing the results via ATOM feeds. So clients can figure out themselves what entries have been updated since the last request.

It is also to be mentioned that the hierarchical structure as implemented in our repository by groups and subgroups, is a simple way to give the users an overview of the offered services and where to look for the requested ones. According to Demian and Fruchter [15] *finding* and *understanding* are the most important performance indicators for a repository, especially for one that includes the UI in the repository itself (through the RESTful paradigm).

5. Related Work

In this chapter we will have a look at other repositories and analyze similarities and difference to the repository introduced in this paper. As pointed out in [16, 17] metadata management covers the following topics:

- Distribution, uniformity of access.
- Metadata should be accessible using service-oriented protocols.
- Management of the metadata lifecycle.
- Granular and uniform access control to metadata.

Comparing the above mentioned concepts our repository covers the following topics:

Distribution, uniformity of access: Metadata of the repository introduced here is available through the RESTful API [4] as shown in listing I. The metadata of the actual services are hosted and maintained by the vendors themselves, only the schema of the metadata is known by the repository. Each client can request the metadata of groups, subgroups, vendors and services. Distribution is covered in section 4.

Metadata should be accessible using service-oriented protocols: We support two ways of accessing service metadata. First, static properties are accessed via downloading a file from the vendors site. Second, dynamic properties are received by issuing a query. All metadata stored in the repository is available through a RESTful interface.

Granular and uniform access control to metadata: The repository also covers a user management for vendors to make sure that only the vendor can change any metadata according to its offered services. General metadata of the repository like the message structures or the properties schema can only be changed by the administrators of the repository.

The topic *Management of Metadata Lifecycle* is not covered by our repository. The client needs to take care about this by itself, or using the techniques associated with Atom [9].

Bernstein et al.[18] point out that a repository should store information about the description of objects and the location of objects. Regarding the repository introduced in this paper both descriptions are stored within the repository and delivered via APP to the client (See List. I). The repository manager defined by Bernstein et al. does not fit the criteria above as there is no version control, notification or work-flow control available. Bernstein et al. conclude that the most important aspect of a repository is a simple interface to interact with the repository. We came to the same conclusion and therefore provide a resource oriented and RESTful API.

6. A Short Overview of the iPhone Application

As mentioned in the introduction we plan to provide an iPhone application that can consume the information and services stored in the repository. There is several building blocks in providing this application:

- The repository holds groups, inside the groups subgroups with the available, substitutable services.
- The properties schema (see Section 3.) has a human-readable description attached. The properties schema is available once per group so a UI for each group of services can be generated automatically. Figure 2 shows an example UI generated from the *ui:caption* annotations shown in List. II. The schema used for generating the UI is also used to check the input collected through the UI.
- The requests to the services are transformed to match the actual service interfaces. Only one way of querying and invoking services has to be implemented on the client side.

A simple workflow is built by connecting the generic descriptions obtained for different groups. Connecting also means that the output of invocations (which is also defined per group) has to be connected to the input of invocations (also defined per group) of other services.

After building the workflow it is saved on the client and can then be executed. Execution means that actual services from the groups are selected and invoked with the inputs, either provided by the user, or collected from subsequent invocations.

Important issues to consider for the application will also be payment and connectivity. Payment may be handled through mechanisms provided by the iPhone itself or gSet [19], a grid based payment method. Connectivity problems, for example being disconnected during running workflows may be solved by storing intermediate results in the repository. It is important for us that the workflow is really controlled from the mobile device, not from the repository, because the mobile device of the user is the assumed secure base (or wallet) of all information necessary to carry out a workflow.

7. Conclusion

In this paper we presented a lightweight RESTful repository to store services that can be utilized in simple user-composed workflows. In order to provide a simple and scalable mechanism to compose these workflows we described how the repository has to be structure, how services can be described by properties, how services can be queried and how simple UI's can be generated from annotations to properties of services.

With this repository as a base, we envision a client for mobile devices, e.g. for the iPhone, that can be utilized by the user to simply connect services, and save his own workflows to use them over and over again. Future work will include performance tests on such a system as well as improved mechanisms for uniform access to RESTful services.

Acknowledgment

This work is partly supported by the Austrian Grid (Phase 2) project funded by the Austrian /bm:bwk (Federal Ministry for Education, Science and Culture), contract GZ BMWF-10.220/0002-II/10/2007.

References

[1] S. Beco, B. Cantalupo, L. Giammarino, N. Matskanis, and M. Surridge, "OWL-WS: a workflow ontology for dynamic grid service composition," in *e-Science and Grid Computing, 2005. First International Conference on*, pp. 8 pp.–155, 2005.

[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2009.

[3] S. Jha, A. Merzky, and G. Fox, "Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes," *Concurrency and Computation: Practice and Experience*, 2009.

[4] L. Richardson and S. Ruby, "RESTful web services," 2007.

[5] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," *Proceedings of Cloud Computing and Its Applications*, 2008.

[6] F. Rosenberg, F. Curbera, M. Duftler, and R. Khalaf, "Composing RESTful services and collaborative workflows: A lightweight approach," *Internet Computing, IEEE*, vol. 12, no. 5, pp. 24–31, 2008.

[7] "Relax NG home page." http://www.relaxng.org/. [Last access: 12.06.2009].

[8] E. Newcomer, *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. Addison-Wesley, 2002.

[9] R. Sayre, "Atom: The standard in syndication," *IEEE Internet Computing*, vol. 9, no. 4, pp. 71–78, 2005.

[10] I. Foser, K. Czajkowski, D. F. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, "Modeling and managing state in distributed systems: The role of OGSI and WSRF," *Proceedings of the IEEE*, vol. 93, no. 3, pp. 604–612, 2005.

[11] L. Bertini, J. C. B. Leite, and D. Mosse, "Optimal dynamic configuration in web server clusters," tech. rep., Technical Report RT-1/08, Instituto de Computação–Universidade Federal Fluminense, 2008.

[12] S. Vinoski, "REST eye for the SOA guy," *Distributed Systems Online, IEEE Internet computing*, 2007.

[13] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs." big"'web services: making the right architectural decision," 2008.

[14] M. B. Blake, A. L. Sliva, M. zur Muehlen, and J. V. Nickerson, "Binding now or binding later: The performance of UDDI registries," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pp. 171c–171c, 2007.

[15] P. Demian and R. Fruchter, "Finding and understanding reusable designs from large hierarchical repositories," *Information Visualization*, vol. 5, no. 1, pp. 28–46, 2006.

[16] O. Corcho, P. Alper, P. Missier, S. Bechhofer, and C. Goble, "Grid metadata management: Requirements and architecture," in *Grid Computing, 2007 8th IEEE/ACM International Conference on*, pp. 97–104, 2007.

[17] P. Missier, P. Alper, O. Corcho, I. Dunlop, and C. Goble, "Requirements and services for metadata management," *IEEE Internet Computing*, vol. 11, no. 5, pp. 17–25, 2007.

[18] P. A. Bernstein and U. Dayal, "An overview of repository technology," in *Proceedings of the International Conference on Very Large Data Bases*, pp. 705–705, INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS (IEEE), 1994.

[19] J. Mangler, C. Witzany, O. Jorns, E. Schikuta, H. Wanek, and I. U. Haq, "Mobile gSET-secure business workflows for Mobile-Grid clients," *Concurrency and Computation: Practice and Experience*, 2009.

Grid Gateway – Accessing grid resources from private networks

Stephan Leitner* and Michael T. Krieger†

Abstract. *Due to the built-in callback functionality of the Globus Toolkit 4 (GT4)it is not possible to access grid resources from clients to a private network behind a firewall using network address translation (NAT) without port forwarding. Several strategies for enabling grid access in such environments exist, ranging from "hole punching" and static port forwarding on the firewall to one central grid access node which holds all grid certificates. To keep the grid certificate on the actual client, to restrict port forwarding to the actual grid proxy lifetime, to free unused but forwarded ports, and to still provide system administrators with control about the actual grid ports, we propose a simple Grid Gateway implementation which enables dynamic port forwarding to grid clients.*

1. Introduction

The infrastructure of the Austrian Grid, the national grid project of Austria funded by the Austrian Federal Ministry of Science and Research (BMWF), is based on the grid middleware Globus Toolkit 4 (GT4). The server components of this middleware use standardized ports for allowing incoming connections to the basic grid services like Gatekeeper, Job Manager, GridFTP, etc. For submitting jobs or file transfer to the grid infrastructure the client typically establishes a connection to the server. But in case of events or output during job execution on the grid, the Job Manager (in the pre web service architecture) or the GRAM service (when using the web service architecture) try to establish a connection to a controllable range of ephemeral ports on the client side.[1]

This communication design works well, if the network interface of the server and the client are located within the public network or the same private network. Allowing this kind of communication between the public and a private network or across networks separated by a firewall requires special modifications either to the grid middleware or the firewall settings. Especially for companies that typically use private network address ranges for the company network and that establish connections to the public network via gateways and firewalls using NAT, the callback to ephemeral ports to grid clients presents a barrier that needs to be overcome when trying to participate in or access a public grid infrastructure.

In this paper we provide a basic overview over the advantages and disadvantages of common techniques used to allow clients behind firewalls using NAT to establish connections to grid infrastructures. We also present the implementation of a Grid Gateway as an improved solution to allow clients from private networks to access public grid services.

*RISC Software GmbH, Softwarepark 35, 4232 Hagenberg, Austria, email: stephan.leitner@risc-software.at
†RISC Software GmbH, Softwarepark 35, 4232 Hagenberg, Austria, email: michael.krieger@risc-software.at

2. Common Strategies for Grid Access from Private Networks

2.1. Central Grid node

If there is no firewall between a central grid node and a client, accessing the grid is trivial. This situation, however, is not given in a typical company environment with a private network, where a client wants to use grid resources from a private network without actually being part of the grid. Even if setting up a grid node within the private network is feasible, the firewall configuration between that node and the rest of the grid might be even more challenging than the configuration needed to connect clients. Nevertheless, this solution might be applicable for larger organizations that want to participate actively in the grid, but is unlikely to be acceptable for clients which only want to access grid resources.

2.2. Static Port forwarding

Port forwarding is necessary as soon as a firewall with NAT exists between a grid node and a client. If done statically, one has to define manually the port range being forwarded or the destination NAT rule for each client that wants to access grid resources. While this might be a suitable solution for a small number of clients within a private network, the following problems have to be considered as soon as multiple clients are used:

- Each client must be assigned a unique port range which must not overlap with the ports used by other clients. While GT4 supports this by using the port range specified as the value of the GLOBUS_TCP_PORT_RANGE environment variable, there is no way to strictly enforce that a client actually uses the ports assigned at the firewall.
- The client's IP address must not change, since port forwarding always forwards packets to one given address.
- Each client has to be added manually.

2.3. "Hole Punching"

The WISENT project presented a possible solution by implementing a technique commonly known as "hole punching". [2] This technique allows to establish a connection over a firewall which uses NAT by using an external broker. Whenever two hosts cannot connect directly, the host initiating the connections (host A) informs the broker which then relays this information to the second host (host B). Host B will try to connect directly to host A but will fail due to host A's firewall. This process will leave an open NAT session in host B's firewall. Host B informs the broker that it has performed the necessary steps and the broker relays this information to host A. Now host A can successfully open a connection to host B, because the firewall will associate the packets with the open NAT session. A more in depth description of this process can be found in [3].

3. The Grid Gateway

While all strategies presented in section 2. have the potential to allow grid access crossing network boundaries using NAT, they all suffer from flaws which we have to address if we want the grid to be fully available to clients located in private networks. The most severe problems are:

- *Flexibility*
  In an ideal system it should be possible for new clients to access grid resources without requiring the network administration to change the firewall configuration manually. This is important for networks with a large number of clients. An arbitrary number of clients with dynamically assigned IP addresses should be supported.

- *Reliability*
  The ability to make new connections should be independent from the firewall implementation employed.

Therefore, we propose a system we call the "Grid Gateway". It consists of a server component responsible for administrating port ranges that can be used for incoming connections and a client component responsible for configuring the client to only use assigned ports. Both components communicate using a socket connection.

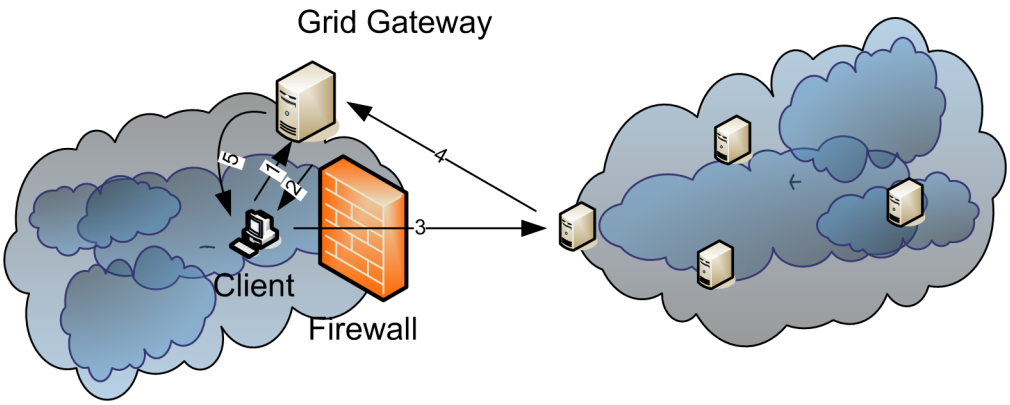


Figure 1. Basic GridGateway steps

The five basic steps to establish a connection as shown in figure 1 are the following:

1. The client requests a free port range.
2. The server creates a destination NAT rule and tells the client which ports to use.
3. The client connects directly to the grid node.
4. The grid node answers to Grid Gateway.
5. The Grid Gateway forwards the packets to the client.

3.1. Grid Gateway Server

The Grid Gateway server runs on a host which possesses a public IP address as well as a network interface exposed to the internal network. It is given a range of ports, which is split up into segments of 50 ports [4]. Each segment can be assigned to a client and belongs solely to that client for a certain time span (lease time). After this lease expires, the port segment is free to be assigned to a different client.

When a client requests a port segment, the Grid Gateway server dynamically adds a destination NAT (DNAT) rule for that segment to the firewall. From that point on, all incoming packets within that segment are forwarded to the client associated with that segment. If the lease expires or the client frees its port segment, the DNAT rule is removed from the firewall's rule set.

3.2. Grid Gateway Client

When a Grid Gateway client is started, it connects to the Grid Gateway server and requests a port segment. If any segments are available, the server will tell the client which ports it can use and the fully qualified domain name (FQDN) of the external IP address. Now the client sets the following environment variables on its local system:

- GLOBUS_HOSTNAME
- GLOBUS_TCP_PORT_RANGE

The grid node sends all call-backs to the Grid Gateway server. Because of the previously dynamically added DNAT rule for this specific host, the Grid Gateway forwards all packets belonging to the call-back properly.

The port segment remains reserved to the client until the client terminates. When this happens, the client sends a message to the server which frees the ports for reuse. Additionally the client sends a heartbeat signal every lease/3 seconds. If no heartbeat is sent, the port segment allocated to the client will also be released. The client can request a new port segment at any time.

4. Conclusion

Although the callback functionality of the Globus Toolkit frees resources on the server and the client side, it also creates barriers for clients from private networks. We presented three common techniques to overcome these barriers and our own solution to the problem of accessing public grid infrastructures from private networks. Still all of the presented solutions require cutbacks on security, which again may raise barriers for organizations considering to participate in national or international grid infrastructures.

References

[1] V. Welch: "FirewallHowTo", *http://dev.globus.org/wiki/FirewallHowTo*, July 2008.

[2] G. Scherp, W. Hasselbring and J. Ploski, "The WISENT Grid Architecture: Coping with Firewalls and NAT", *German e-Science Conference 2007*, Baden-Baden, Mai 2007.

[3] B. Ford, P. Srisuresh and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators", *USENIX Annual Technical Conference*, 2005

[4] G. L. Volpato and C. Grimm, "Empfehlungen zur statischen Konfiguration von Firewalls im D-Grid", *D-Grid Integrationsprojekt technical report*, 2006.

Development of a Special Method and a Software System for the Semi-automatic Segmentation of Biplane Angiograms

Bernhard Quatember, Martin Mayr, Wolfgang Recheis[1]
Stefanos Demertzis[2]
Giampietro Allasia, Roberto Cavoretto, Alessandra De Rossi, Ezio Venturino[3]

During the past decades, much effort has been devoted to achieving an improved understanding of the contractions of the heart. Numerous methods have been developed and applied to capture specific data from medical images which are then used to describe the motion of the heart. However, the accuracy of these methods is somewhat restricted. Thus, we aimed at an improved method of motion tracking. The pivotal point of our development efforts is the full exploitation of the synergy of the two imaging modalities, viz. cardiac CT and biplane cineangiography, for an accurate quantitative assessment of the regional variations of ventricular motility and to monitor the improvements during therapy. The extraordinarily high spatial and temporal resolution of biplane cineangiography facilitates accurate investigations of the ventricular motility, which, however, are restricted to the regions of the left ventricular surface that are covered by the coronary arteries. Several methods to extend the tracking of the ventricular motion to the entire epicardial surface have been described in the literature. However, the shape of the epicardial surface was only an assumed one and thus not patient-specific. Our multimodal imaging approach to motion analysis is based on a retrospectively ECG-gated cardiac CT data set at the end of the diastole and a biplane cineangiogram of a particular patient. The utilization of an efficient combination of both imaging modalities enables us to exploit their specific favorable characteristics of both of them and to overcome their respective limitations. In particular, we fully take advantage of

- the capacity of cardiac CT to retrospectively reconstruct a highly-accurate 3D image of the epicardial surface at the end of the diastole and
- the excellent motion tracking capability of biplane angiography, in which there is no significant motion blur.

This paper is confined to the generation of a deformable mesh to represent the geometry of the outer surface of the myocardium and its deformations during the cardiac cycle which are highly complex

[1]Innsbruck Medical University (Radiology), Anichstrasse 35, 6020 Innsbruck, Austria
[2]Cardiocentro Ticino, Via Tesserete 48, 6900 Lugano, Switzerland
[3]Universita degli Studi di Torino (Matematica), Via Carlo Alberto 10, 10123 Torino, Italy

and difficult to describe quantitatively. Our long-term goal, however, is to exploit our mesh generation approach to develop methods to elucidate the spatial and temporal changes of strain and stress within the myocardium of the rapidly moving heart.

In the CT imagery for the end-diastolic position, we carry out a segmentation of the surface of the myocardium. In each transaxial slice, we define manually an appropriate number of vertices and a spline (NURBS) curve which approximates these vertices. These splines are subdivided into equal increments. The subdivision points of all transaxial slices are regarded as nodes of a surface mesh. The nodes belonging to each transaxial slice are connected to each other with an interpolating spline (NURBS curve) which is called transversal contour curve. We thus obtain an array of k NURBS curves as transversal contour curves of the outer surface of the myocardium. We also compute an array of longitudinal interpolating NURBS curves, each of which passes through corresponding points in all slices. These two arrays of NURBS curves constitute a surface mesh which enables us to achieve a preliminary visual representation of the heart wall (myocardium, left ventricle) for the end-diastolic position in the form of a wireframe model.

We then accomplish a landmark-based image co-registration

- of this mesh belonging to the CT data set for the end-diastolic position and
- the biplane angiograms for the end-diastolic position within a cineangiograhic sequence taken from the same patient (frame rate about 30 to 60 frames/s).

In doing so, we use selected bifurcation points of the coronary arteries in both modalities as anatomical landmark points. This co-registration is based on thin plate spine transformation. Our motion-tracking and analysis procedures of the aforementioned (initial) mesh are also based on a number of thin-plate spline transformations that refer to corresponding bifurcation points of the coronary arteries in the individual frames of the cineangiographic imagery.

Figure 1(a) shows a transaxial slice of the CT data set; Figures 1(b) and 1(c) show both projections of the biplane angiograms. The figures refer to the end-diastolic position.

Figure 2 illustrates the co-registration of our two imaging modalities, viz. cardiac CT and biplane angiography.

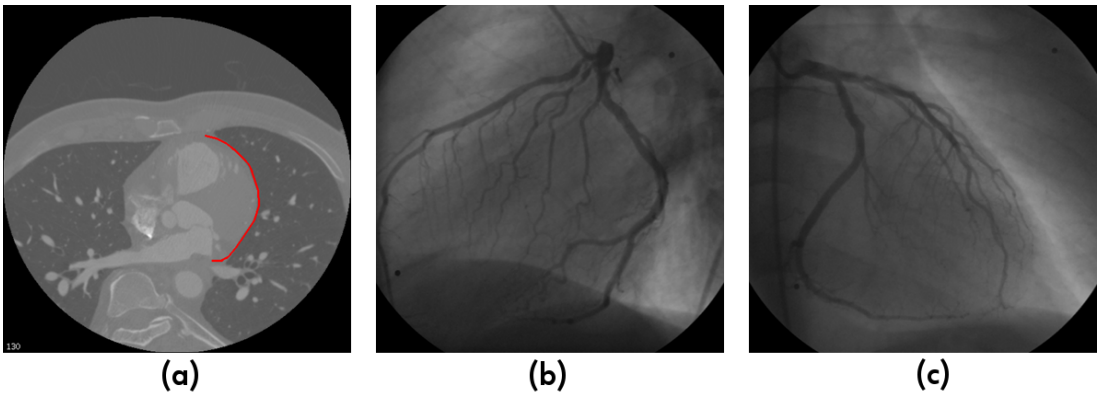


Figure 1. Medical imagery relating to end-diastolic phase: (a)transaxial slice of CT with manually segmented ventricular wall; (b)+(c)biplane angiograms relating to end-diastolic frame of cineangiography

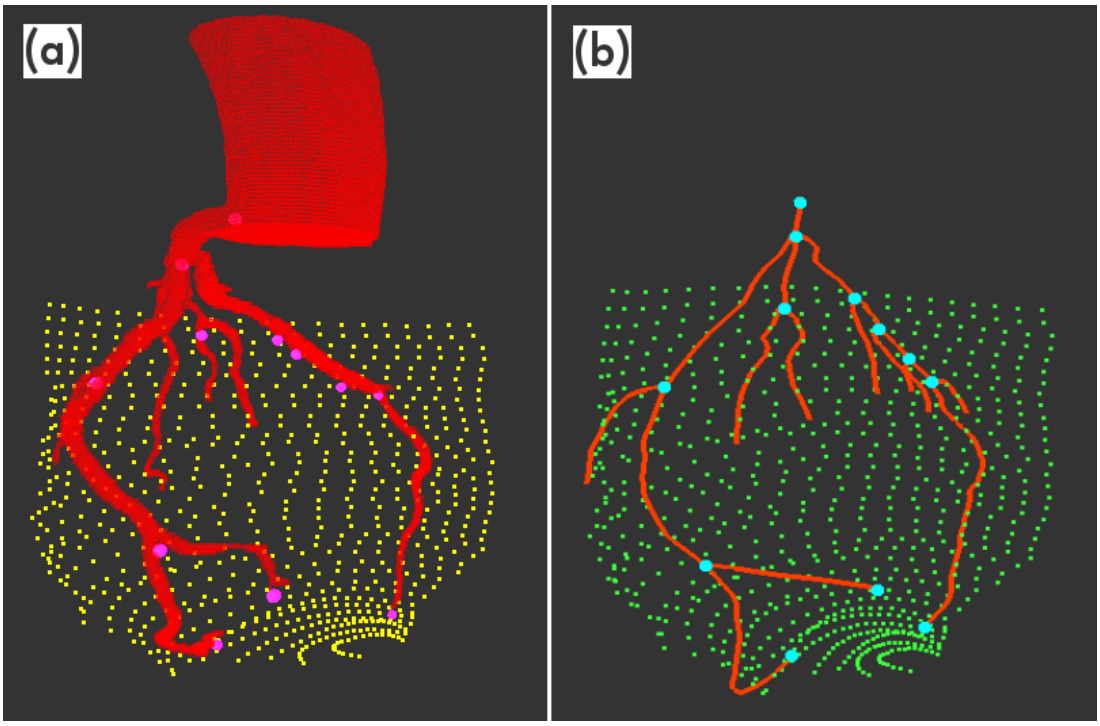


Figure 2. Image Registration: (a)ventricular surface (nodes of the mesh) after segmentation in CT, point landmarks are indicated in magenta; (b)ventricular surface after registration to the 3D representation of the coronary artery tree belonging to the end-diastolic phase, point landmarks are indicated in cyan

In Figure 3, the heart wall deformation between diastole and systole can be seen.

By using our methods for motion analysis, it will be possible to quantitatively assess the regional motility of the epicardial surface of the left ventricle with an especially high accuracy which makes investigations even of small improvements of the regional motility over time feasible. This high-performance capacity of our method is highly important to the field of cell therapy of coronary artery disease.

We aim to achieve a GRID implementation within the framework of the Austrian GRID, an initiative supported by the Austrian Federal Government. The above-described procedures are computationally expensive, since we need skeletonized representations of the three-dimensionally reconstructed coronary artery trees. Representations of this kind must be calculated for all frames of the cineangiogram. A cineangiogram contains typically 60 frames belonging to 60 equally spaced points in time during an entire cardiac cycle. As have been described earlier (especially in [1, 2]), the generation of each skeletonized coronary (epicardial) artery tree requires computationally highly expensive procedures that must be carried out in both angiograms (projections) belonging to a particular frame of the biplane cineangiogram. But, fortunately, the processing of the 60 frames can be done in parallel, since the computations can be subdivided into numerous complex, but largely independent tasks that can be computed in parallel. This coarse-grain parallelism is well suited for a utilization of the clusters of workstations incorporated into the Austrian GRID. As has also been described in [1] in detail, the exploitation of this parallelism leads to a considerable enhancement of performance (speedup of about 12-13 after Amdahl's law). Moreover, a further slight increase of this impressive speedup can be achieved by the applied interleaved processing of the above-described thin-plate spline transformations and the constructions of the Gordon surfaces. Beyond that, the utilization of the Austrian GRID

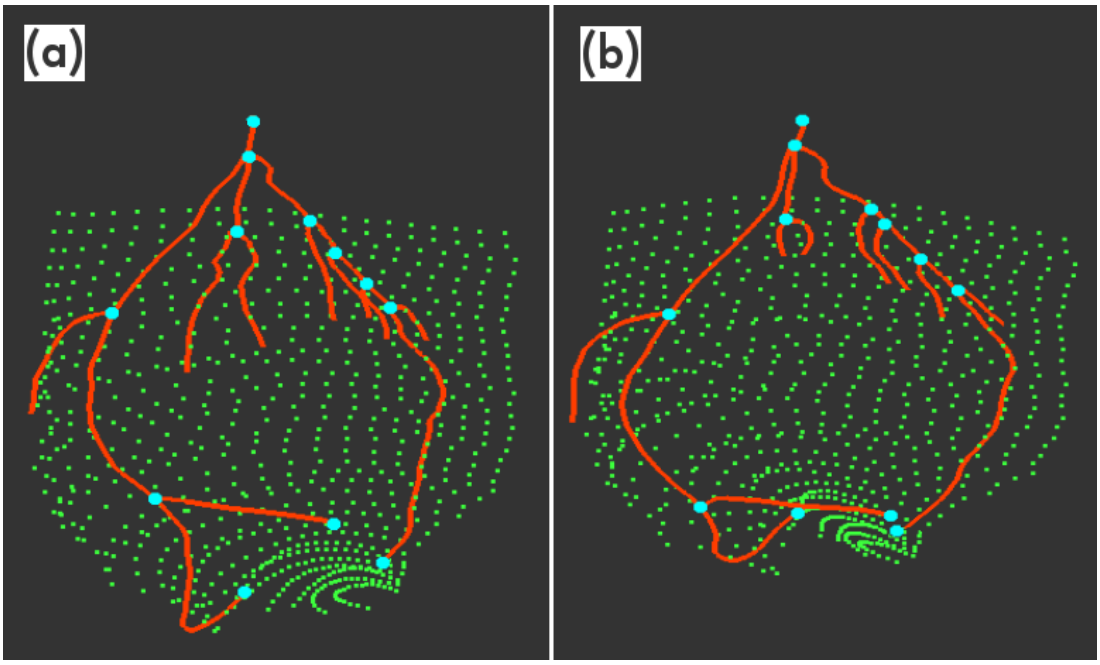


Figure 3. Comparison between ventricular surface (nodes of the mesh) during end-diastolic phase and at systole: (a)end-diastolic phase, point landmarks are indicated in cyan; (b)systole, point landmarks are indicated in cyan

allows us to yield a further advantage, since cardiologists distributed throughout a geographical area will have full and equal access to the developed facilities [1].

Acknowledgements

The work described in this paper is partially supported by the "Austrian GRID" project, funded by the Austrian BMBWK (Federal Ministry for Education, Science and Culture) under contract GZ 4003/2-VI/4c/2004./

References

[1] B. Quatember, M. Mayr, and H. Muehlthaler. Clinical usefulness of a computational grid for diagnosis and planning therapy of coronary artery disease. In J. Volkert, T. Fahringer, D. Kranzlmller, and W. Schreiner, editors, *1st Austrian Grid Symposium*, volume 210, pages 75–89, Schloss Hagenberg, Austria, 2006. Oesterreichische Computer Gesellschaft (books@ocg.at).

[2] T. Fahringer, R. Prodan, B. Trawoeger, B. Quatember, and M. Mayr. Workflow modelling of grid-based system for diagnosis of coronary artery disease with agwl. In J. Volkert, T. Fahringer, D. Kranzlmueller, and W. Schreiner, editors, *1st Austrian Grid Symposium*, volume 210, pages 234–244), Schloss Hagenberg, Austria, 2006. Oesterreichische Computer Gesellschaft (books@ocg.at).

Assuring Grid Accesses in the g-Eclipse Project

Jie Tao and Mathias Stuempert*

Abstract. *This paper presents our work in assuring the functionality of the g-Eclipse project which provides a general framework for grid users, operators, and application developers to access and manage the grid resources. We achieve this goal mainly with two mechanisms: verifying the functionality with testing and ensuring the code quality with metrics. This extended abstract gives an overview of the first scheme which grants the access to the existing grid platforms.*

1. Introduction

g-Eclipse [2, 3] aims at providing a generic framework that allows users to access the power of the existing grid infrastructures via a standardized, customizable, and intuitive interface. This framework is designed for all grid users, operators, and application developers. Grid users can interact with grid resources in a simple, graphical way without having to know the technical details. For example, files can be transferred across grid sites by drag&drop. Resource providers can use the intuitive tools to operate and maintain the grid sites, manage the virtual organizations, and perform benchmarking. Application developers reduce the development cycle with the g-Eclipse support of remote building and deployment tools.

g-Eclipse is developed on top and is an official project of Eclipse [5]. It can be expected that g-Eclipse will acquire increasing users. Hence, quality of this software must be guaranteed. The quality control of g-Eclipse covers various aspects, including code quality, documentation quality, and program style. This paper focuses on the concept and feature of unit testing in a grid environment, an initiative work in the fields of both grid computing and software quality assurance (SQA).

A specific feature of g-Eclipse test is the verification of grid functionality. Different from algorithmic tests of single units, which are usually only related to the program codes, grid tests involve complicated grid processes like authentication, VO specification, etc. These actions are actually done at the runtime using wizards of the g-Eclipse environment, which are not available for the off-line tests. Hence, we have developed mechanisms to automate all of these processes. For a comprehensive validation, we designed workflows to simulate the real use cases.

2. Junit Test and the Eclipse Plug-in Test

Testing is an efficient way to verify the functional requirement of a unit. For object-oriented programming, testing is performed by test classes comprised of test methods. A test method represents a test case

*Steinbuch Centre for Computing, Forschungszentrum Karlsruhe, Karlsruhe Institute of Technology, Postfach 3640, 76021 Karlsruhe, email: {*jie.tao, mathias.stuempert*}*@iwr.fzk.de*

which validates the results of the method under test by calling it with typical input. Many companies provide tools to support unit tests.

Junit [1, 6] is a simple, useful testing framework for Java. It defines how to structure the test cases and provides a test runner to run them. Eclipse not only integrated this test runner into its platform but also implemented a PDE test runner, an extension of the Junit one, for running plug-in tests.

3. Hierarchical Testing Concept and Implementation

For fully verifying the expected functionality of g-Eclipse, we designed a hierarchical testing concept with basis tests, grid tests, and integration tests using realistic workflows.

Basis test. Basis tests are typical unit test that verifies single methods. In this case, only one method is involved in the test. Depending on whether the Eclipse runtime is required, theses tests are executed either with the Junit test runner which only launches the user's workspace or the PDE test runner that also starts Eclipse to initialize the variables the tested method relies on.

Grid test. g-Eclipse is a platform for users to access the existing grid infrastructures. It supports all basic grid activities like job creation and submission, file transfers, job/resource monitoring, remote debugging, application deployment, benchmarking, etc. It is important to have a stable, maintainable implementation of these functionalities. Hence, a large part of the testing work was conducted with the verification of the Grid functionality and the establishment of a testing framework for this.

Grid testing is more complicated than the basis tests: it requires a procedure for authentication, VO setting, project creation, etc. This procedure is actually conducted by the g-Eclipse user via wizards at the runtime which are not available for the static tests on the Eclipse development environment.

Hence, we designed an automation process and developed a stub class implementing all required initialization for a grid activity. Based on the stubs, Grid functionality can be tested. Again, it is a hard work to bring the Grid test into work. First, it is difficult to pre-know which stub has to be included in the test case: one test may only need a single stub while another may require all of them. The reason is that most of the tests are not written by the developer of the original code. Hence, the test developers are not familiar with the functionality of the classes under testing. The same reason results in the other difficulty: to involve associated methods in the test case. Methods for grid operation often have dependencies. For example, to activate a token must first validate it. We achieved this knowledge through debugging.

Integration test. The integration tests are designed to test the interfaces between units. While unit tests address individual modules, integration testing focuses on how the modules fit together. This is necessary to grant the functionality of the system as a whole, especially when it is established by many developers, like the case in g-Eclipse.

We define workflows which perform a certain task with various functionalities involved. The test class is written to follow the workflow and thereby deploys all required methods. Theses methods may be found in one class, within a package, or even from different plug-ins. A sample workflow is demonstrated in Fig. 1.

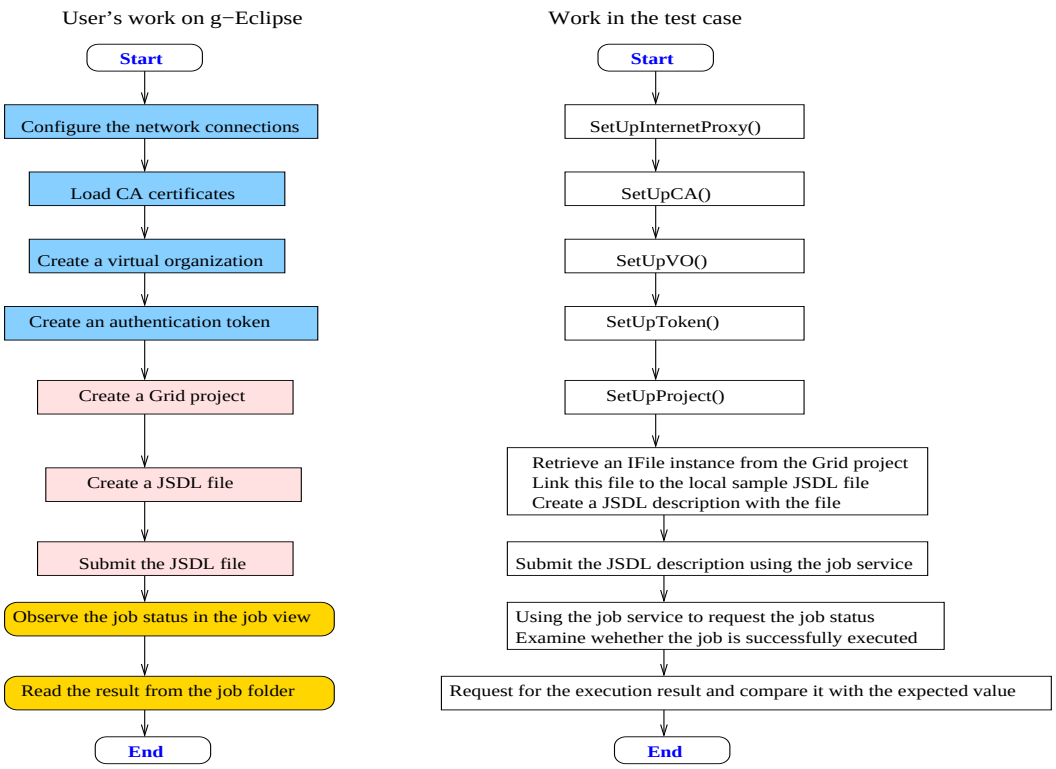


Figure 1. A workflow of submitting a Grid job (left: using g-Eclipse, right: with testing).

The left side of Fig. 1 presents the work of a user who is intending to try the job submission functionality using g-Eclipse, while the right side of the figure shows the corresponding work in a test case which simulates the user's activities and thereby inspects whether the system behaves correctly.

The user starts with several initialization tasks. The first task is configuring the network connection to grant that g-Eclipse on the user machine can reach the resources and services on the grid. The second task is to load the CA certificates for authenticating the servers. The other two tasks, creating a VO and a token, is needed to identify and authenticate the user.

After the initial preparation, the user can now generate a Grid project, where several folders are combined. One of these folders is the Grid Job Description Folder which holds the job descriptions to be created by the user. Another folder is the Grid Job Folder which maintains an entry for each submitted job to store the job information and the execution result. Hence, the next task of the user is to create a job description file for his computing. For this, g-Eclipse offers a wizard where users can specify the required parameters. The job description file is automatically generated. Now, users can submit the job by selecting the "Job Submission" item from the menu list.

In the following, the job status is presented in the job view. In this view, users can see in which step the job is being processed. If a job fails, the reason will be shown. In case of a successful job submission, the execution result is downloaded and stored in the Grid Job Folder.

Now we observe how the user's actions are described in the test case. As shown in the right side of Fig. 1, the testing starts with the Grid initialization. It uses the stubs to conduct the activities users perform

on the g-Eclipse platform. Similarly, the Grid project is also created with the corresponding stub. For job description, a sample file is prepared and stored on the local disk. In the test case, this file is linked to an IFile instance of g-Eclipse and a job description object is created based on this file instance. Using the "submitJob" method of the job service class the job can be finally submitted. The job service class also provides methods for requesting the job status. For this testing, we update the job status every minute until the status can not be changed, i.e. "aborted", "done", or "cleared". If the job is successfully executed and the Exit Code is zero, the job result is downloaded. Failure is reported to the user through the assertion statements.

Currently, g-Eclipse supports gLite [4] and GRIA [7] based grid infrastructures. We have run the texts on resources provided by the EGEE project and the project partners. Relying on the failure type we distinguish the errors of g-Eclipse from those caused by the grids.

In summary, the integration testing models the real grid accesses of the user, involves various code modules, and therefore is capable of detecting system bugs that do not occur in simple unit tests. We have designed and simulated a set of such workflows for verifying the basic grid functionalities of g-Eclipse.

4. Conclusion

g-Eclipse provides a platform for users to access the Grid. It has to be granted that the platform behaves correctly with various actions of different users. This work developed a test infrastructure to verify the promised functionality of the project, thereby assuring the access path from users to the grid infrastructure. As the tests are automatically run at a daily base, bugs and problems can be removed.

References

[1] K. Beck and E. Gamma. Test Infected: Programmers Love Writing Tests. *Java Report*, 3(7), July 1998.

[2] H. Kornmayer et al. gEclipse- An Integrated, Grid Enabled Workbench Tool for Grid Application Users, Grid Developers and Grid Operators based on the Eclipse Platform. In *Proceedings of the 2nd Austrian Grid Symposium*, Innsbruck, Austria, September 2006.

[3] P. Wolniewicz et al. Accessing Grid computing resources with g-Eclipse platform. *Computational Methods in Science and Technologie*, 13(2):131–141, 2007.

[4] S. Burke et al. gLite 3.1 User Guide. available at http://gnuplot.info/docs/gnuplot.htm.

[5] E. Gamma and K. Beck. *Contributing To Eclipse: Principles, Patterns, And Plug-Ins* . Addison-Wesley Professional, 2003.

[6] Junit.org. Resources for Test Driven Development. available at http://www.junit.org.

[7] M. Surridge, A. Taylor, D. De Roure, and E. Zaluska. Experiences With GRIA-Industrial Applications on a Web Services Grid. In *Proceedings of the First IEEE International Conference on e-Science and Grid Computing*, pages 98–105, 2005.

Token Based Authentication Methods for the Grid

Thomas Ludescher*, Willy Weisz†, Marco Descher‡, Thomas Feilhauer*

Abstract. *This paper describes the advantages and possibilities of using secure tokens for authentication, encryption, and decryption within a Grid environment. To use secure tokens within the Austrian Grid infrastructure, the necessary changes to the Public Key Infrastructure (PKI), its security policy, and to the Certification Authority operations will be discussed. After the evaluation and selection of the secure token type (SmartCard or USB token) to be used for the authentication and identification process of persons, the necessary software changes will be implemented for the Austrian Grid. This affects the application initiating the cryptographic operation, as well as the library changes needed to enable the use of secure tokens. As an extension we plan support for TPM within the Globus container which means that our final implementation is going to provide secure token usage for both users and hosts within the Austrian Grid infrastructure. To show the whole security concept a secure MyProxy server allowing remote attestation will be implemented.*

1. Introduction & Motivation

Token-based authentication is consistently gaining ground within IT. Several entities support or rely on card based authentication, e.g. e-Government, financial sectors, e-card.

Secure tokens protect the secret part of cryptographic keys by storing it in a way that allows its use in secure operations performed on the token's processor only and prevents the export of information that would enable the reconstruction of the secrets outside the token's physical confinement.

Using token based two-factor authentication (physically presenting the only and generally encrypted instance of the secret or private key, and knowing how to decrypt it, i.e. the PIN), prevents the situation where the owner of the cryptographic keys is unaware of the fact, that someone else is potentially in the possession of his secrets. Additionally the limited number of PIN trials allowed before the token becomes unusable makes a brute-force attack on the encrypted key impossible [2].

2. Policy

What policy changes would be deduced from replacing/augmenting the current key structure with token based authentication (i.e. SmartCard / TPM)?

*Research Center Process- and Product-Engineering, Fachhochschule Vorarlberg, Dornbirn, Austria, email: {thomas.ludescher | thomas.feilhauer}@fhv.at

†Department of Scientific Computing, Universität Wien, Vienna, Austria, email: willy.weisz@univie.ac.at

‡Information & Software Engineering Group, Vienna University of Technology, Vienna, Austria, email: marco.descher@tuwien.ac.at

Figure 1 shows a possible way of how to create a secure token based certificate. The key pair will be created on the card and the private key will never leave the card during its whole life cycle. The public key in the form of a certificate signing request (CSR) will be output by the token and submitted to the signing Certification Authority. The issued certificate that will in turn be made available on the token, will also certify that the associated private key is securely stored in an enclosure it can never leave. The procedure will be documented in the CP/CPS document of the CA.

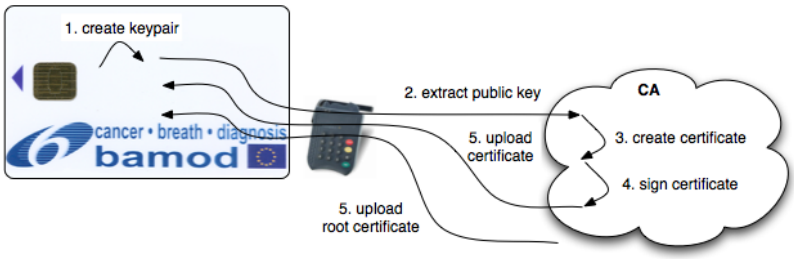


Figure 1. Personalize token

3. Token Based Authentication

Token-based authentication offers the cryptographic security of a Public Key Infrastructure for the identification (through the subjectName of the certificate) and the authentication (through the cryptographic key pair), and the secure storage for the private key which cannot be extracted and copied.

These advantages are available for authenticating both peoples and hosts, the respective technologies, however, differ. We will elaborate on the use of SmartCards for user authentication and Trusted Platform Modules (TPM) for host authentication.

3.1. SmartCard/Token

For token based authentication several different hardware types can be used. In this extended abstract smartcards and USB tokens are presented.

A Smart Card, chip card or integrated circuit card (ICC), provides embedded integrated circuits on a pocket-sized card that can process data. There are two major categories of ICCs ([1], [2]). Memory cards contain only non-volatile memory storage components and perhaps some specific security logic. Microprocessor cards contain non-volatile memory and microprocessor components. For our purpose a microprocessor card is required. Figure 2 shows the smart card that is already in use within the BAMOD project.

To communicate with the microprocessor a card reader is required. Figure 3 shows a card reader with pinpad from ReinerSCT.

USB tokens incorporate the same type of chip as smartcards, thus providing the same on-chip security features. They combine a smart card with a built-in USB card reader without pinpad in a very small enclosure, providing two-factor authentication, digital signature, and encryption capabilities in a portable format. IDpendant[1] provides a USB token as shown in figure 4.

[1] http://www.idpendant.de/produkte/usb-token.html

Figure 2. BAMOD - SmartCard

Figure 3. ReinerSCT card reader with pinpad

Figure 4. USB token from IDpendant

The common benefits of both Smart Cards and USB tokens are:

- They can be used as secure data storage.
- It is not possible to use the secrets on the card without knowing the valid PIN nor to extract the private key.

They differ in the off-chip security:

- The SmartCard reader with a pinpad offers the extra security of a Trusted Path because the PIN is entered without passing through the computerbusses, memory and processor, and thus cannot be spied out by key loggers.
- Card readers may also incorporate a finger print reader using the latter for generating the PIN or adding a third factor to the authentication.
- The advantage of the card reader integrated into the small USB token form factor is paid for by the lack of a secure path for the PIN entry.

3.2. Trusted Platform Module (TPM)

The Trusted Platform Module (TPM) is an additional hardware component soldered to the mainboard of the computer system. The functionality of a TPM is related to that of a SmartCard with the main difference that it is firmly bound to a specific hardware, e.g. the mainboard of a computer system. This enables an unambiguous identification of the respective computer system.

Among other duties (e.g. true random number generation, SHA-1 engine), TPM provides hardware methods to create and securely store RSA keys. There are two methods of how to obtain a security certificate that uses a TPM-protected RSA private key:

- Create a Trusted Computing System (TCS) RSA key pair and get a certificate for the secured keys from an accredited CA.
- Add TPM protection to a RSA private key stored on disk, and for which an X.509 certificate exists.

A TPM's capabilities could hence be used to enhance the current method of storing public/private key pairs on the respective host system.

3.3. Implementation Details

The Austrian Grid CA will extend its services to provide higher valued certificates for cryptographic key pairs generated – under the supervision of one of its RAs (Registration Authorites) – in secure tokens in such a way that their private keys cannot be extracted.

Austrian Grid's WP11 implements the secure-token resp. TPM functionality for the corresponding authentication methods required for use in the Austrian Grid infrastructure. We decided to use a SmartCard from a vendor that provides a SmartCard operating system with PKCS#11 interface for different operating systems. As a first test we decided to use Siemens CardOS V4.3 B tokens with the Siemens CardAP V3.2I. Additional solutions from other manufacturers will be used in further tests. With this provided tools an existing Austrian Grid certificate could be easily stored on the SmartCard for test purposes. In order to test the cryptographic tokens stored on the card the PKCS#11 interface was used within the email client (Thunderbird) and the web browser (Firefox). All tests passed successfully. The actual Java CoG Kit Library does not support the PKCS#11 interface. To support the new Sun PKCS#11 provider[2] an additional class (org.globus.pkcs11.tools.PKCS11ProxyInit) was added and tested. With these small changes the `grid-proxy-init` command supports all tokens with a compatible PKCS#11 interface.

4. Conclusion and Future Work

An infrastructure with security tokens, such as smart card or USB-tokens, is better protected against identity theft and key loggers than cryptographic keys stored on conventional mass storage [2].

There will also be the need for the definition of procedures to do the identity vetting of TPM tokens similar to those already in place for individuals. But what should be the TPM equivalent of a passport or ID card issued by a public administration authority?

In the future we will provide an adapted `cog-myproxy` command. This function provides main functions for retrieving, removing, and storing credentials on MyProxy server. It also provides functions for getting credential information and changing passwords. Additionally we will modify the Globus Toolkit Middleware to support the respective certificate stored within the TPM to accommodate for the host authentication.

To show the full advantages of the security concept a secure MyProxy server will be provided. The final version will fulfill the following requirements (a) only users with tokens can use the myProxy server, (b) the server attests to the client, and (c) the server stores the proxy encrypted with the TPM public key to be bound on the machine. This concept needs two software parts (1) a Java applet that is signed with the a certified key of the CA and is hosted at the austriangridca.at server and (2) the MyProxy server components.

References

[1] W. Rankl. *Chipkarten-Anwendungen*. Hanser Fachbuch, 2006.

[2] W. Rankl and W.Effing. *Handbuch der Chipkarten*. Hanser Fachbuch, 2002.

[2] http://java.sun.com/javase/6/docs/guides/security/p11guide.html

Software execution constraints for correctness checking of workflows

Maximilian Berger* and Romedius Weiss† and Thomas Fahringer‡

Abstract. *Grid workflow execution is not always successful: Differences in installation and configuration may results in the failure of individual activities, which may not be notices until the workflow completes. In this paper we present a simple constraint system with pre- and postconditions which can be used to verify correctness during the execution, thus allowing the detection of failures earlier, when corrective action can still be taken.*

1. Introduction

Modern scientific experiments are run in highly distributed environments such as the Grid [4] or the Cloud [6]. Computational jobs are not executed locally, but on a remote site, which is likely to be under the control of a different administrative domain. The execution environment is heterogeneous: It consists of different operating systems, different software stacks, and different versions of libraries. For an application developer, it is difficult to adapt an application to support all possible combinations of hardware, software, and system libraries. As such, an application may fail or produce incorrect results when executed on a remote site, whereas the same execution completes successful when run locally.

Failures in the execution may not be discovered immediately. Larger applications consist of a workflow of single activities. Each activity takes a given input and produces a result. The workflow specifies which data is then passed to which new activity. If a result is incorrect, the next activity may fail, or worse not detect the faulty data, and also produce an incorrect result.

We try to solve this problem by adding constraints defining pre- and postconditions for each activity to be run in the distributed environment. The constraints will be specified using a well-defined language and checked during the execution, immediately preceding and following an execution.

The paper is organized as follows: in section 2. we describe the general approach we are investigating. Section 3. gives a glimpse at the verification of the work. Section 4. lists some related work. Finally the paper is concluded with conclusions in section 5..

*Distributed and Parallel Systems, University of Innsbruck, Austria; email: maximilian.berger@uibk.ac.at
†email: romedius.weiss@student.uibk.ac.at
‡email: tf@dps.uibk.ac.at

2. Approach

In most existing systems, individual activities are submitted to a remote site by an execution engine. In some cases, the activity has to be wrapped to execute additional functionality. We will extend the activity wrapper to include the constraint checking mechanism.

An activity wrapper will be submitted instead of the original activity. The activity wrapper has several inputs: It takes the original activity as an input, the original activity arguments, and new data for the activity wrapper itself. In the case of the constraint system this additional data consist of the pre- and postconditions. Another additional data is the constraint checker itself, if it is not installed on the remote site.

The constraint checker is split into two parts: A parser and an executor engine. While the parser is executed locally, e.g. in the execution engine, the executor is run on the remote site to verify the constraints.

2.1. Constraint Language

The constraint language is based on a simplified Java expression syntax. It consists of three different syntactic elements: primitives, functions, and operators. Its main design decision is simplicity: Other constraint languages fail to be used because they are too complex. We wanted something that every user will be willingly to type in, so we chose simple strings describing the constraints. Figure 1 gives an example.

```
pre  = fileexists('input.pov')
post = fileexists('output.tif') && filesize('output.tif') > 128
```

Figure 1. Example constraints

The primitives include numbers, boolean, and strings values. Just like the Java language, these data types are not interchangeable. While all three data types can be used as arguments to functions, a function may return either a boolean or a number. Operators include comparison operators and simple mathematical for numbers and boolean operators for the boolean types. The final value of a condition must be a boolean.

Functions are used to actually evaluate if a condition is met. Functions may either return boolean or numerical values. A function may take any number of parameters, which may be primitives or expressions. An example would be a *fileexists(string)* function which would check if a given file exists on the remote system. Another example is a *filesize(string)* function which would return the size of a given file. Internally, functions are mapped to program code, so they may provide any functionality.

We are also supporting "unknown" as a value for both numerical and boolean variables. Since the constraint is parsed and executed in two different components, the executer may not have all functions available yet. If a function cannot be mapped, it will return the value unknown. This value is then used in the expression. If a constraint results in unknown, it is assumed to be valid. This will result in a check which is not done. However, as the other checks are still run, the result will still be better than without the constraint system. Furthermore, this allows independent updates of parser and executor.

2.2. Parser and Executor

The parser and executor are split into two components. The main rationale behind this is correctness checking: If the execution engine parses the given constraints, syntax errors can be detected before the activity is submitted to the remote site. Another reason is performance: parsing the constraint and creating the internal representation takes some time, which can be saved on the remote site.

The executor checks the constraints by mapping the functions to implemented functions and evaluating the constraints. If the execution environment supports verbosity, it may return an extended status describing which tests succeeded and which failed. If the execution environment does not support this, it will just return "fail", which will then be interpreted as a failure of the activity. This failure can then be picked up by the execution engine and be used to re-submit the activity.

3. Verification

As the work discussed here is still in progress the verification plan is described. We plan to verify this software in at least two Grid installations, with at least two different workflows.

First we will verify the solution in the Austrian Grid [1]. We will add the constraint software to the ASKALON [3] tool system, which is tested and well used in the Austrian Grid environment. ASKALON allows us to run multiple workflows from different domains.

We will also verify the solution in the EGEE [2] production Grid. Unlike the Austrian Grid, which is mainly used for research, the EGEE Grid is used for production. Here, issues such as differing software installation and configuration appear more often.

For the workflow to be used, we plan to use both a simple example of POV-Ray and the more complex workflow of Wien2k. Both applications have been successfully ported to the Grid in previous works. However, due to misconfiguration on remote Grid sites, they sometimes fail.

The POV-Ray workflow describes a simple image rendering task: An image rendering scene is split up into several sub-pictures. Each sub-picture is then submitted to a different site for execution. At the end, a complete picture is built by combining the tiles to a complete picture. If POV-Ray is not installed correctly, it fails to create the output file. Until now, this is not noticed until the combination task is run.

The Wien2k workflow is an example from computational chemistry. It uses the linear augmented plane wave method to calculate properties of crystals. When the execution of a Wien2k activity fails an output file is created, but in addition an `error.log` file is created with the detailed message. Currently, the existence of this error file may be unnoticed until the very end of the workflow, where an incorrect result is produced.

4. Related Work

Constraint checking for Grid execution environments has been implemented in the Otho Toolkit [5]. This implementation differs in two aspects: First, the language for Otho was based on special characters, such as $\exists$, which are not readily available on the keyboard to be typed. Seconds, previous work

does not include the "unknown" value, thus not supporting different versions of parser and executor components.

Any system can be extended to simple constraint checking by adding the constraint checks to the execution script. This solution is currently used in the execution of the Wien2k workflow. However, this approach lacks the variability of the approach presented in this paper.

5. Conclusions

In this paper we have outlined a simple, yet powerful constraint system which can be used for checking pre- and postconditions for Grid activities. The language described is simple enough so that it will be attractive to use in Grid application development.

The constraint checker will be integrated into the existing ASKALON environment, making it instantly available to all users of ASKALON in the Austrian Grid. The components itself are easily portable to other Grid and execution environments.

What remains to be checked is the amount of overhead which will be generated from evaluating the constraints. Since the boolean expressions are simple, preliminary results show the execution time to be in the range of seconds, whereas most activities run for minutes, thus creating a neglectable overhead.

References

[1] Austrian Grid. `http://www.austriangrid.at/`.

[2] Enabling grids for E-sciencE (EGEE). `http://www.eu-egee.org/`.

[3] Thomas Fahringer, Alexandru Jugravu, Sabri Pllana, Radu Prodan, Clovis Seragiotto Junior, and Hong-Linh Truong. ASKALON: A Tool Set for Cluster and Grid Computing. *Concurrency and Computation: Practice and Experience*, 17(2-4), 2005. `http://www.askalon.org`.

[4] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers, second edition, 2004.

[5] J. Hofer and T. Fahringer. The Otho Toolkit–Synthesizing tailor-made scientific grid application wrapper services. *Multiagent and Grid Systems*, 3(3):281–298, 2007.

[6] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. Technical report, January 2009. `http://ccr.sigcomm.org/online/?q=node/439`.

Providing Semantic Data Integration as a Grid Service

Thomas Leitner, Andreas Langegger, and Wolfram Wöß*

Abstract. *Especially for scientific collaboration, sharing data between different institutions is very important. In order to facilitate this task, the Grid-enabled Semantic Data Access Middleware (G-SDAM) has been developed, which is capable of integrating distributed, heterogeneous data sources in grid environments. The core component of G-SDAM is SemWIQ (Semantic Web Integrator and Query Engine), which can be used independently as well for Web-based semantic data integration.*

In this contribution, the architecture and functionality of G-SDAM is described. G-SDAM integrates SemWIQ into existing grid middleware such as Globus Toolkit 4 by means of a grid service. Beside the description of the architecture and the implementation of G-SDAM, the integration of authorization based on Virtual Organizations is discussed.

1. Introduction

While several projects towards semantic data integration exist in general, typical data grid middleware is not capable of integrating heterogeneous data sources based on formal mappings and semantics. The aim of the G-SDAM project (*Grid-enabled Semantic Data Access Middleware*) is to enable virtual semantic data integration in grid environments. It consists basically of two parts, the *Semantic Web Integrator and Query Engine* (SemWIQ) and the grid service that enables its use in grid environments with the grid middleware Globus Toolkit [6].

During the last years, valuable research has been done in the Semantic Web community addressing issues of Web-scale semantic data integration. SemWIQ, as a dedicated information integration system based on Semantic Web concepts, combines data sources that are located at different endpoints to a holistic virtual data source. To increase the benefit of the SemWIQ project, it has been released independently from G-SDAM. The key aspect is to integrate existing heterogeneous data sources which are autonomous and geographically distributed such as archives and scientific data repositories. SemWIQ is based on a mediator-wrapper architecture [7] and has been described in detail in [4]. In order to query different data sources, it is necessary for them to register at the SemWIQ mediator. It is possible for clients to establish a connection to the mediator and to run queries against it. The system uses the *Resource Description Framework* (RDF), which is the core data model of the Semantic Web, as the global data model in support of describing data based on domain ontologies. The integrated data is translated into RDF triples as part of the integration process. Because the virtual global data source is built as a virtual RDF graph (Resource Description Framework), the query language used is SPARQL[1].

*Institute for Application-Oriented Knowledge Processing (FAW), Johannes Kepler University, Altenberger Strasse 69, Linz, Upper Austria, Austria, email: {thomas.leitner|al|wolfram.woess}@jku.at

[1]SPARQL is an acronym for the *SPARQL Protocol and RDF Query Language*, which defines a query language for RDF graphs as well as a HTTP REST-based protocol.

2. Related Work

Typical *data grid* middleware, such as the *Storage Resource Broker* [5], does not support virtual data integration based on formal mappings. Globus Toolkit, which has been chosen for the G-SDAM project, as well as other current grid middleware implementations, do not provide a functionality to virtually integrate distributed, heterogeneous data sources based on a formal mapping and addressing semantics of data. OGSA-DAI (*Open Grid Services Architecture Data Access and Integration*) [2], which enables grid-based data exchange and OGSA-DQP (*OGSA Distributed Query Processing*) [1], which supports distributed query processing, both do neither address semantics nor the specification of formal mappings to address the heterogeneity aspect.

Concerning the virtual integration of heterogeneous data sources, the authors of the *AutoMed* integration system [8] have addressed grid integration. There are several projects in the area of semantic web services, which are based on dynamic service composition [3]. However, there is currently no database-centric approach which uses query federation and distributed query processing in a similar way to G-SDAM.

3. Integration of SemWIQ into Globus

Before the architecture of G-SDAM is explained, the SemWIQ core system is described in short. The entire system is depicted in Figure 1. Query processing in SemWIQ is based on the federation of global SPARQL queries. The process is described in short as follows.

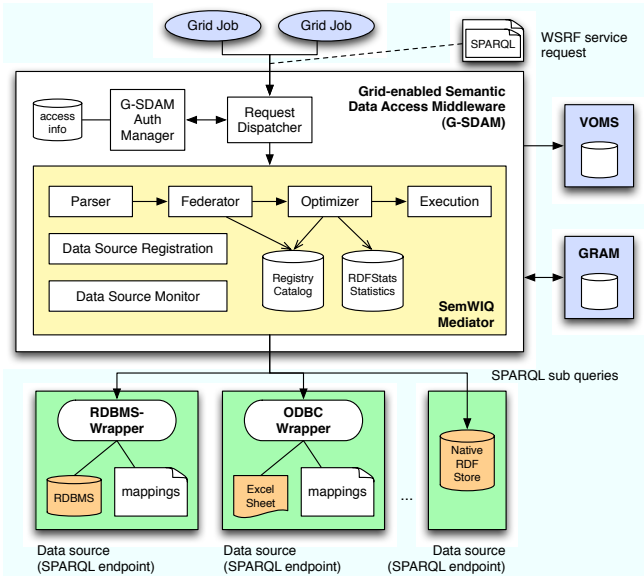


Figure 1. Architecture of G-SDAM with SemWIQ mediator.

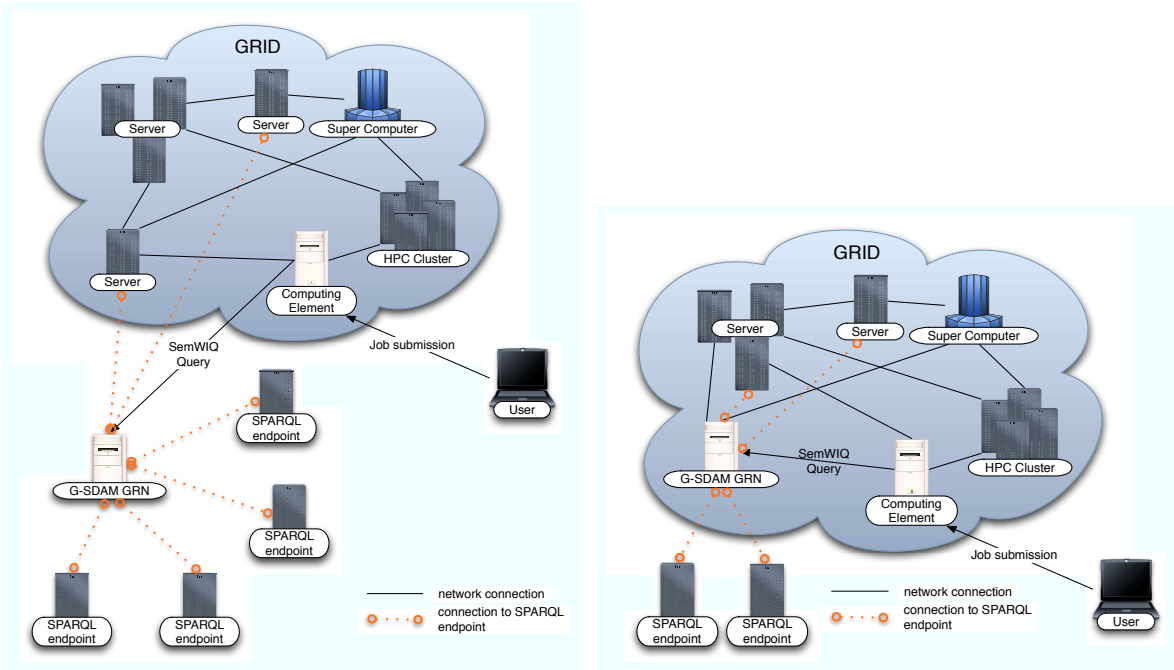
The SPARQL parser calculates a canonical query plan which is modified by the federator and afterwards by the optimizer component. The federator searches the registry and statistics for relevant data sources and builds a federated query plan while the optimizer generates an optimized query plan using rules that are based on heuristics and statistics. These statistics are refreshed periodically. The query execution engine processes the optimized query plan by transferring sub-plans to wrappers of remote data sources and evaluating the remaining global plan. The whole query execution process is

based on operator pipelining and enables the streaming of query results to the client.

To enable semantic data integration in grid environments, G-SDAM wraps the SemWIQ mediator, which becomes accessible through a grid service. The component which is hosting the grid service and the SemWIQ mediator is called *Global Repository Node* (GRN) in the G-SDAM context. It provides services for data source registration and query execution.

3.1. Approaches of Grid Integration

Since grid technology has its own middleware to communicate with resources in an efficient and comfortable way, different possibilities to best integrate SemWIQ have been investigated. Because there are several conditions that have to be considered, different ways of integration with specific benefits and challenges are described. Basically two operating modes can be distinguished. In the first one, depicted in Figure 2(a), the GRN is located outside of the grid infrastructure and in the second one, shown in Figure 2(b), SemWIQ is deployed on a grid node and thus it is part of the grid infrastructure. At the moment, a SemWIQ mediator which is located outside of the grid can already be used by grid jobs. Regarding the second mode, the full integration of SemWIQ into Globus Toolkit 4 is currently being developed. A first prototype is already available.



(a) SemWIQ outside of the grid

(b) SemWIQ inside of the grid

Figure 2. Integration of SemWIQ into Globus Toolkit

3.2. SemWIQ as a Grid Service

In order to support SemWIQ as part of the grid middleware (second mode), it has been integrated as a grid service. The GRN, where the SemWIQ mediator is located, is not deployed on a resource outside of the grid but on a grid node. Thus, it is fully integrated in the grid infrastructure and no further authentication mechanisms between other grid resources and the GRN are necessary. The grid resource simply sends the job to the GRN where the SemWIQ mediator processes the SPARQL query.

Then the result is streamed back to the user. The key challenge of this approach is the deployment of SemWIQ into existing grid middleware including the security infrastructure, resource addressing, and allocation.

Since Globus Toolkit 4 it is possible to develop so-called grid services which are basically web services. Because of the fact that simple web services are stateless, they are not well suited for asynchronously executed grid jobs. Grid services enhance ordinary web services with a state which is stored in a concept called *resource*. The principle of binding this resource to a web service is defined by the *Web Service Resource Framework* (WSRF). The state of a web service is stored as values of pre-defined attributes. In order to maintain the state the resource concept stores these attributes. As defined in WSRF the web service URI gets an unique identifier to be able to reference a specific resource. This concept is called *WS-Resource* in WSRF.

In order to integrate SemWIQ into Globus Toolkit, a grid service has been implemented which has a server-side implementation for the communication between the grid middleware and SemWIQ. Client side stubs can be used as interfaces by users in an appropriate way for gaining access to the semantic data integration system.

3.3. Support for Virtual Organizations

Because the concept of virtual organizations is very powerful concerning the benefits gained in security and authorization aspects, this concept will also be supported for authorization with SemWIQ. Since every user is registered in one or more virtual organizations, it is reasonable that he/she gets only access to data sources that share at least one *Virtual Organization* (VO). If a data source is located outside of the grid but registered at the SemWIQ mediator, it is possible for any user to gain access to this information, because such data sources do not need any privileges (only access to the SemWIQ middleware is necessary). The key aspect of this approach is that the privileges of the user are determined and sent to the SemWIQ mediator. The mediator only selects those data sources available to the user based on the corresponding privileges. This is done by contacting a *Virtual Organization Membership Service* (VOMS), which stores the information concerning VOs.

4. Conclusion

Currently, there exists no grid middleware supporting the virtual integration of distributed, heterogeneous data sources based on Semantic Web concepts. In order to establish SemWIQ in grid environments, the G-SDAM project has been initiated. Two different modes have been discussed in this paper including the usage of SemWIQ as a service outside of the grid and its full integration as a grid service. Depending on the integration mode, it is necessary either to provide an additional authentication and authorization mechanism or to integrate the existing *Grid Security Infrastructure* (GSI) and contact a VOMS server to fetch the necessary VO information to determine the authorization of the user. By autumn 2009, G-SDAM will be available as a service for the users of the Austrian Grid. On the one hand, users may provide their data sources by registering them at the G-SDAM *Global Repository Node* (GRN) and on the other hand they can access the integrated information from grid jobs via the GRN.

Acknowledgments

RDFStats is part of G-SDAM and funded by the Austrian BMBWK, contract GZ BMWF-10.220/0002-II/10/2007.

References

[1] M. Nedim Alpdemir, Arijit Mukherjee, Norman W. Paton, Paul Watson, Alvaro A.A. Fernandes, Anastasios Gounaris, and Jim Smith. Service-based distributed querying on the grid. In *Proceedings of the 1st International Conference on Service Oriented Computing*, pages 467–482. Springer, 2003.

[2] Mario Antonioletti, Malcolm Atkinson, Rob Baxter, Andrew Borley, Neil P. Chue Hong, Brian Collins, Neil Hardman, Alastair C. Hume, Alan Knox, Mike Jackson, Amy Krause, Simon Laws, James Magowan, Norman W. Paton, Dave Pearson, Tom Sugden, Paul Watson, and Martin Westhead. The design and implementation of grid database services in ogsa-dai: Research articles. *Concurr. Comput. : Pract. Exper.*, 17(2-4):357–376, 2005.

[3] Ontology Engineering Group. OntoGrid. `http://www.ontogrid.net/ontogrid/index.html`, May 2009. Last visit: June 2009.

[4] Andreas Langegger and Wolfram Wöß. SemWIQ – Semantic Web Integrator and Query Engine. In Heinz-Gerd Hegering, Axel Lehmann, Hans-Jürgen Ohlbach, and Christian Scheideler, editors, *Beiträge der 38. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, volume 1. Bonner Köllen Verlag, 2008.

[5] R.W. Moore, M. Wan, and A. Rajasekar. Storage resource broker; generic software infrastructure for managing globally distributed data. *International Symposium on Mass Storage Systems and Technology*, 0:65–69, 2005.

[6] B. Sotomayor and L. Childers. *Globus Toolkit 4: Programming Java Services*. Morgan Kaufmann, San Francisco, CA, 2006.

[7] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[8] Lucas Zamboulis, Hao Fan, Khalid Belhajjame, Jennifer A. Siepen, Andrew C. Jones, Nigel J. Martin, Alexandra Poulovassilis, Simon J. Hubbard, Suzanne M. Embury, and Norman W. Paton. Data access and integration in the ispider proteomics grid. In *DILS*, pages 3–18, 2006.

Ray Tracing in Grid Environments

Thomas Odaker and Jens Volkert*

Abstract. *Ray Tracing is an image generation technique that requires a significant amount of computational processing to render almost photo-realistic images. This paper focuses on porting the basic algorithm to a Grid Environment thus speeding up the rendering process and on the compensation for the bottlenecks in performance and network bandwith that appear within the Grid thus not loosing the peformance gain offered by the number of nodes the Grid provides.*

1. Introduction

With the evolution of computational Grids towards a valuable research platform that can process massive amounts of data the demand for visualization of the resulting data rises.
Visualization helps scientists to analyze, debug and evaluate their data allowing them to see connections or phenomena that are not clearly visible from the raw data.

Ray tracing was chosen as the preferred technique for visualization due to its ability to visualize a wide range of data (from volume data to triangle based scenes) and the possibility to run it on parallel nodes quite easily.
In fact ray tracing has been well researched and commonly used on shared memory machines as well as clusters to visualize large amounts of scientific data, render near-photorealistic images and create entire movies. [3]
Implementing the algorithm on those machines is a well-researched area. [2]
The advantage of these machines is their uniformity which makes it relatively easy to distribute data, make good use of the available resources and achieve maximum rendering performance.

However, a computational Grid poses several challenges due to its topology when ray tracing on the nodes of the Grid. The wide variety of nodes makes it difficult to distribute the work among the individual nodes while balancing the load evenly. In addition the possibly limited network bandwith can pose as a serious bottleneck that can vastly decrease the overall performance of the application thus almost cancelling the speedup achieved by the use of the computational Grid.

The main obstacles when ray tracing on the Grid are:

Load balancing Due to the wide variety of nodes and the upredictable system load of these nodes a near optimal distribution of the necessary calculations is very difficult.

Data distribution Distribution of the data among the nodes has to be done with system resources and network bandwith in mind.

*Institute of Graphics and Parallel Processing, Johannes Kepler University, Linz, Upper Austria, Austria

Network bandwith Since there is no guaranteed network bandwith between the nodes a tradeoff between the time needed for calculations and transfer of the resulting data has to be found.

2. Rendering Process

The basic architecture that is used for rendering is based on a single server that manages a number of computational resources which perform the calculations.

The server is a single machine that is controlled by the user. Initially all data needed for the rendering process is stored on this machine.
The worker nodes connect to this server which handles the data distribution among the nodes and distributes the work with the goal to balance the load despite a possible diversity of the computational nodes.

The rendering process can be split into three individual steps:

- Scene distribution
- Rendering
- Transfer of resulting data

2.1. Scene distribution

The scene distribution is the initial data transfer of the entire scene description to all the nodes as well as the update of the scene between two frames.
Since it is not predictable which ray will hit which objects of the scene each node has to store the entire scene. Thus the first step is to transfer the scene to all the grid resources before starting the rendering process. [1]

If multiple consecutive frames are rendered the dynamic parts of the scene need to be updated between two frames. A full redistribution of the entire scene would be very impractical at this point. Instead incremental updates are transmitted and applied to the already received scene data.

Due to possibly very limited bandwith of some nodes and/or the server this step is crucial to the perfomance of the entire rendering process. If a node cannot offer enough bandwith to transfer the entire scene data in a reasonable amount of time it is not useful for the rendering process and should thus be eliminated from the rendering process not to tie up bandwith or resources and achieve a higher overall performance.

2.2. Rendering

The second step is the rendering of the resulting image.
It is quite obvious that the optimal perfomance in this step would be achieved by having each node calculating one part of the resulting image with the same computational time on each node thus

reducing the waiting time and network overhead to a bare minimum.
The algorithm chosen to approximate this behaviour is to break the entire resulting image into a number of smaller areas. The smallest area possible would be a single pixel that is calculated by tracing a single ray of light. Distributing the image based on single rays would however cause massive communication, tie up lots of resources and bandwith, thus greatly reducing the overall perfomance.

The areas chosen for distribution should be reasonably large. Distributing too large jobs however could slow down the overall rendering performance since some nodes may have less processing power or high load. In this case the faster nodes would be idle while the server would still be waiting for the missing data from the slower nodes.
Choosing the size of one area for optimal perfomance proves to be very difficult and is depending on the scene and the nodes available.
Another approach is to make this size variable and have the server choose it at runtime depending on previous rendering times or remaining data to be processed.

2.3. Transfer of resulting data

The last issue is the retransfer of the resulting data back to the server which is responsible for assembling the different parts into a single image.

In order to reduce overhead a single block of data is formed. Benchmarks have shown that usually the most efficient way to do this is to use the same render areas for computation as well as the transfer back to the server.

Another important point is to do this by integrating the endianess conversion and composition of the package into the rendering process itself rather than into the transfer process. This way the idle time of the network connection during the rendering process can be used to transfer the data back to the server without loosing additional time for the data transfer.

3. Results and Future Work

3.1. Results

Preliminary benchmarks have shown a speedup factor of up to 23 (depending on the scene rendered) for the rendering and retransfer stage of the rendering process compared to a local multi-core workstation. The initial scene distribution however still causes a great decrease in performance and is target for future work.

Small network bandwith however has a serious performance impact due to the long time intervals needed for distribution of the scene and retransfer of the resulting image and cancels out a lot of the performance advantage gained by the usage of the computational grid.

Another issue that came up is the problem of system load as single machines with very heavy load caused massive performance drops in the overall rendering performance that need to be compensated for.

3.2. Future Work

Future work will investigate the possibilities to improve performance by varying the size of the distributed render areas based on system load, network load and available resources of the worker nodes and improving algorithms for determining bottlenecks and scaling the render areas accordingly.

Another field of work is the scene transfer to be distributed among nodes with fast connections rather than immediate rendering thus creating a hybrid system of nodes distributing data to elements with slower connections and rendering nodes and thus creating a multi stage distribution of the data.

4. Acknowledment

This work was supported by the Austrian Grid Project funded by the Austrian Federal Ministry of Science and Research under contract GZ BMWF-10.220/0002-II/10/2007. The results discussed in this work were achieved using grid resources provided by Austrian Grid.

References

[1] Wilfrid LEFER, ”An Efficient Parallel Ray Tracing Scheme for Distributed Memory Parallel Computers”, *Proceedings of the 1993 Symposium on Parallel rendering*, p 77 - 80, 1993.

[2] Ingo Wald and Carsten Benthin and Philipp Slusallek, ”Distributed Interactive Ray Tracing of Dynamic Scenes”, *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, p 11, 2003.

[3] Robert L. Cook and Thomas Porter and Loren Carpenter, ”Distributed Ray Tracing”, *ACM SIGGRAPH Computer Graphics Volume 18, Issue 3*, p 137 - 145, 1984.

Developing Astrophysics Workflow Applications with the ASKALON Environment in the Austrian Grid

Simon Ostermann*, Markus Brejla†, Radu Prodan*
Thomas Fahringer* and Sabine Schindler‡

Abstract. *The Grid is a powerful tool for science coming at the cost of a higher complexity compared to desktop, workstation or cluster systems. The ASKALON system has been developed to allow scientists to work with Grids in an easy to use fashion. The complexity is masked and hidden as much as possible, and only application deployers need to know which resources are available and the details that are important for their software. Creation of a workflow and its execution can be done without any Grid domain specific knowledge with the ASKALON toolkit once the atomic application parts are registered in the system. Two real world astrophysics applications, which have long runtimes and potential to be executed in parallel, have been used to demonstrate the power of the system and evaluate the performance and efficiency achieved using the computing infrastructure from the Austrian Grid.*

1. Introduction

The Austrian Grid offers heterogeneous computational resources to the participating Universities to solve their complex problems on this shared distributed system. Most of the applications used by scientists are not in a Grid compatible form and therefore need to be ported to this special kind of infrastructure to allow parallel executions. The ASKALON system [2] offers a way to abstract the complexity of this resource class and shields the user from the technical Grid related details. An application needs to be registered with the resource management system. The user then can start an operation using a system-independent graphical tool (using Java Webstart technology) to build a workflow out of the registered atomic application parts. The execution of this workflow, including all input and output parameters, is managed by the same graphical interface to allow user-friendly interface. In this work we will present two astrophysics applications that require a massive amount of computation power to deliver results in a reasonable timeframe. The ASKALON system is used to execute and monitor [3] the runs of this application in a massively parallel scale running on hundreds of CPUs in the Grid. The executions are then analyzed and optimized to improve execution performance. The resulting application allows the astrophysics to execute their simulation in a way as simple as they are used to but enhanced with the power of the Grid.

The paper is organized in the following way: In Section 2 the two applications are introduced and their workflow structure is shown. Additional sections about porting the two applications to the Grid and the changes required will be added in the full paper. Experimental runs using different amount

*Distributed and Parallel Systems, University of Innsbruck, email: simon@dps.uibk.ac.at
†University of Innsbruck, email: markus.brejla@student.uibk.ac.at
‡Institute for Astro- and Particle Physics, University of Innsbruck, email: astro@uibk.ac.at

of Grid resources and multiple problem sizes will be evaluated. The obtained speedup and efficiency of the different executions will be shown for each problem size per workflow. Additional related work will be listed and compared with the presented approaches. The comparison with related work will lead to future work that might be required to optimize ASKALON. The paper will end with a conclusion about the current state of the system and its usability.

2. Applications

In this section we give details about the two astrophysics applications that were ported to the Austrian Grid and will be evaluated in the final version of this paper. Montage is the workflow used by other Grid workflow systems [1] and the porting to ASKALON will allow us to do a better comparison. Grasil is an application used by the Institute of Astrophysics in Innsbruck in cooperation with the Distributed and Parallel systems group. The goal for this cooperation was to allow the astrophysics to perform larger simulations by minimizing the runtime of this application.

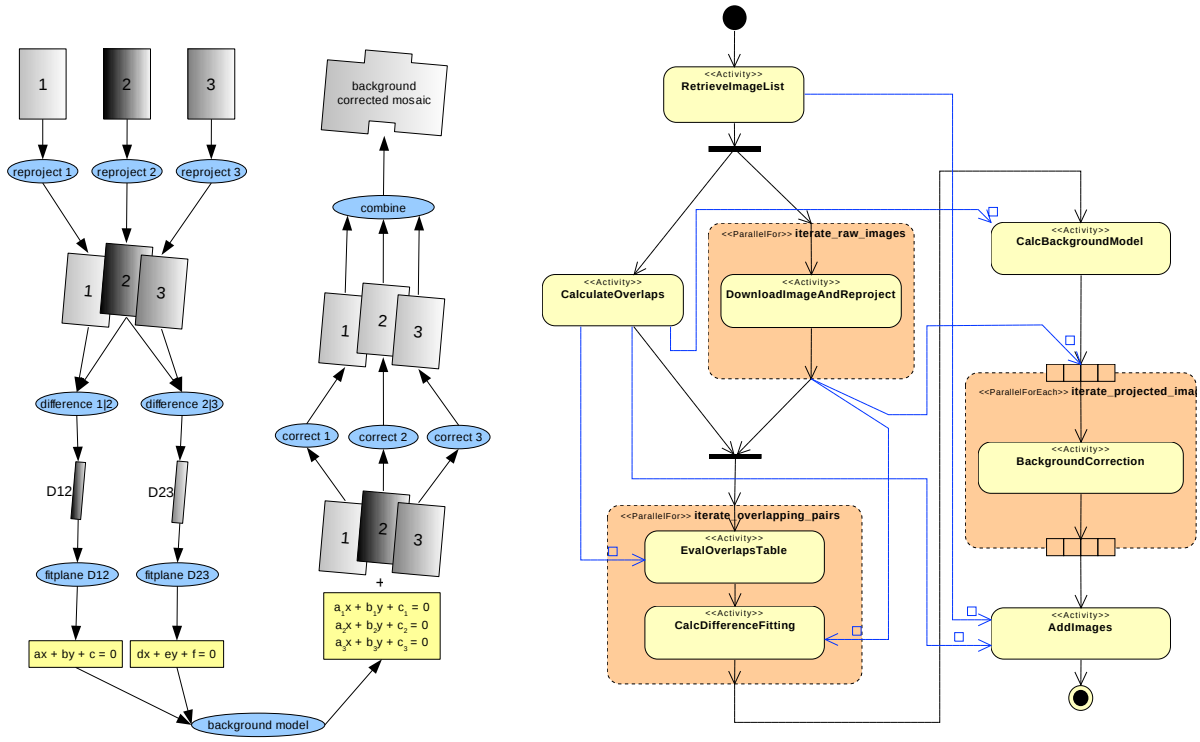
2.1. Montage

Montage [1] is a toolkit for assembling astronomical images into combined image mosaics. The toolkit is maintained by the NASA/IPAC Infrared Science Archive and can be used to study large portions of the sky that could not be viewed otherwise as the area covered by a single images is too small. The toolkit consists of a set of command line tools providing functions for reprojecting, correcting and coadding images as well as some utilities for gathering input images and handling large image files. The Montage workflow translates the single steps of the mosaic creation process into an application that can be executed on the ASKALON Grid computing environment. The workflow makes use of the Montage tools and executes them in parallel wherever possible.

Multiple astronomical surveys provide imagery of the sky. When a telescope makes an image the celestial sphere is projected to the flat picture plane, and there are many possible mappings to achieve this. In order to federate images from different sources and projections, a standard data format is used called Flexible Image Transport System (FITS). FITS files support header keywords that provide descriptive information about the data such as photometric and calibration information, along with the origin of the data. In astronomy several standardized World Coordinate Systems (WCS) are used that make it possible to map each pixel of the image to a location in the sky.

When merging multiple astronomical images into a single mosaic several operations have to be performed to get a correct and uniform result. Each input picture is rotated, stretched and projected to a common viewpoint. In order to smooth out the different background levels a uniform background model is calculated for the complete mosaic and corrections are applied to all images. Reprojecting images is a very time consuming operation, fortunately one that can be parallelized by computing the reprojection for each input picture at the same time. This is where the main potential for speedup lies within the Montage workflow, but there are more steps that can be effectively executed in parallel. The steps in the Montage workflow are:

- Reproject images
- Create difference images and calculate coefficients for background correction
- Calculate global background model



(a) The schematic workflow showing the processing of the images.

(b) The UML representation of the workflow in ASKALON.

Figure 1. The Montage application.

- Perform background correction on reprojected images
- Combine corrected images into a single mosaic

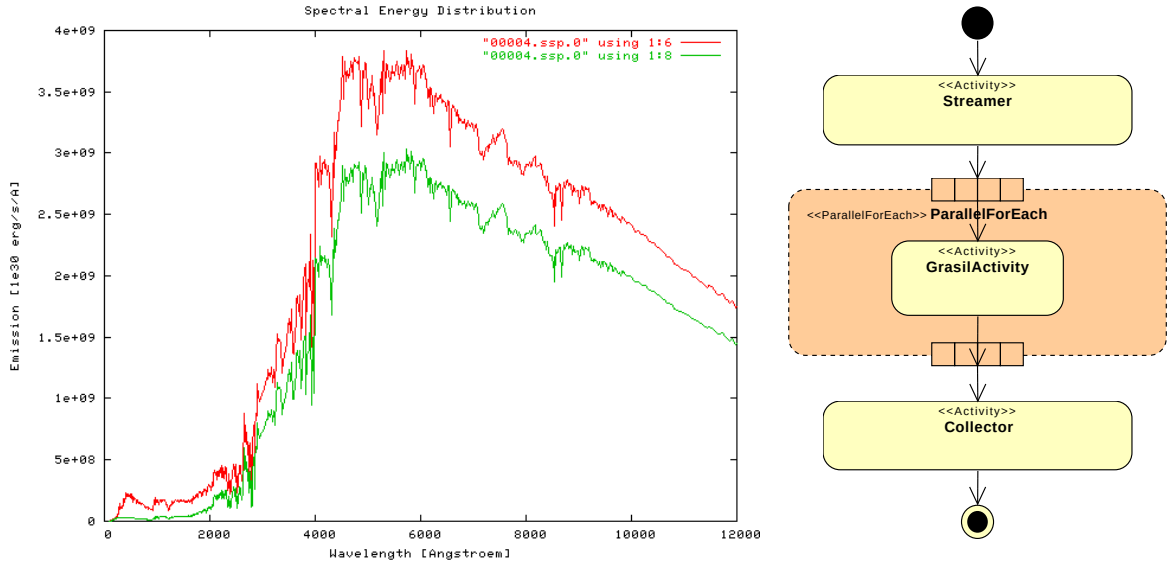
Figure 1(a) shows the different steps of this application in a schematic way for a mosaic built from three images. Figure 1(b) shows the workflow developed in ASKALON using a workflow composition tool UML where the three orange boxes represent the parallel sections.

2.2. Grasil

Dust is an indicator of star formation. Knowledge about star formation leads to knowledge about galaxy formation, which leads to knowledge about the very structure of our universe. Usually, this dust emits infrared light. Young, far away galaxies though, (high redshift) show their dust in the sub-mm band.

Currently, there is a new generation of sub-mm telescopes being built. To predict what they will see (and to interpret the results, once available), it is necessary to consider the effects of dust. GRASIL [4] is a FORTRAN code that can compute the spectral energy distribution of a galaxy taking into account the effects of dust. This code has been tailored for high redshift galaxies at the Institute of Astrophysics in Innsbruck.

Since these calculations are computationally intensive, calculating the SED (spectral energy distribution) for hundreds of thousands of independent galaxies is an ideal application for the Grid.



(a) Spectral Energy Distribution of a Galaxy as calculated by Grasil (green: contribution of dust).

(b) The UML representation of the workflow in ASKALON.

Figure 2. The Grasil application.

The Institute of Astrophysics in Innsbruck wants to calculate the spectrums for thousands to millions artificial generated galaxies to build a collection of this data for a later matching of real recorded spectrums from space. This catalog of results will help to categorize newly discovered galaxies.

Figure 2(a) shows one possible output of a spectral energy distribution generated with this application from a galaxy. Figure 2(b) shows the workflow we developed for the Grid considering a parallel section surrounded by two sequential activties.

The full version of the paper will continue here with the sections: Porting to the Grid, Evaluation, Related Work, Conclusion and Future Work.

References

[1] G B Berriman, J C Good, A C Laity, A Bergou, J Jacob, D S Katz, E Deelman, C Kesselman, G Singh MH Su, and R Williams. Montage a grid enabled image mosaic service for the national virtual observatory. In *ASP Conference Series Vol XXX*, 2004.

[2] Thomas Fahringer, Radu Prodan, Rubing Duan, Francesco Nerieri, Stefan Podlipnig, Jun Qin, Mumtaz Siddiqui, Hong Linh Truong, Alex Villazón, and Marek Wieczorek. Askalon: a grid application development and computing environment. In *6th IEEE/ACM International Conference on Grid Computing (GRID 2005), November 13-14, 2005, Seattle, Washington, USA, Proceedings*, pages 122–131. IEEE, 2005.

[3] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, Thomas Fahringer, and Alexandru Iosup. Workflow monitoring and analysis tool for ASKALON. In *Grid and Services Evolution*, pages 73–86, Barcelona, Spain, 2008.

[4] Laura Silva, Gian Luigi Granato, Alessandro Bressan, and Luigi Danese. Modelling the effects of dust on galactic seds from the uv to the millimeter band. *The Astroph. Journal*, 509:103, 1998.

Running Hierarchical N-Body Simulations on Grid Infrastructures

Paul Heinzlreiter, Jens Volkert*

Abstract. *The computational N-body problem represents a well-known problem of computational science, which computes the evolution of large amounts of particles under the the influence of mutual gravitational attraction. Due the quadratic complexity of the problem hierarchical methods delivering approximate results such as the Barnes-Hut algorithm have been developed. Parallel implementations for shared and distributed memory parallel architectures have been done regardless of the inherent irregularity and dynamicity of the data distribution and communication patterns. This paper describes an hierarchical N-body implementation based on the Barnes-Hut method being able to run on a heterogenous grid infrastructure consisting of different machines such as desktop workstations, SMP clusters, and shared memory supercomputers.*

1. Introduction

The basic idea of a gravitational N-body algorithm is to simulate the movements of a large number of particles being induced by their mutual gravitational attraction over several timesteps. The gravitational force between two particles can be calculated using the vector form of Newton's law of universal gravitation. During the simulation the force vector which is exerted on one particle is calculated for all combinations with other particles and the results are accumulated. After the acceleration induced by the force vector has been calculated based on the particles mass a numerical integration method is applied to determine the velocity and the new position of the particle at the end of the timestep. The computational complexity of the N-body algorithm is dominated by the force computation which requires $O(N^2)$ force computations for N particles. The integration steps leading to the new position of the particle can be calculated in $O(N)$ steps and require no further interaction among the particles.

To alleviate the quadratic complexity of the force computation hierarchical algorithms like the Barnes-Hut (BH) [1] method have been developed. The basic idea of these algorithms is given by the observation that the amount of gravitational interaction is inversely proportional to the square distance between two particles. Based on this observation the force being exerted can be approximated as follows: Distant particles, which are collocated within a small volume compared to the distance to point of space onto which their impact should be calculated, can be approximated by a single particle positioned at the clusters center and containing the summed mass of the particles it represents. In case of the BH algorithm the accuracy of the approximation can be steered using an opening angle

*GUP Linz, Johannes Kepler University Linz, Altenbergerstraße 69, 4040 Linz, Austria, email: heinzlreiter@gup.jku.at

parameter θ.

$$d > \frac{l}{\theta} \tag{1}$$

If this criterion holds the particle cluster sized l is sufficient for calculating the force being exerted on a particle located at a distance d away from the clusters mass center. Otherwise its accuracy is insufficient and the subclusters or particles contained need to be evaluated seperately. The accuracy of the BH method is steered by the parameter θ. The force evaluation using the BH method results in a computational complexity of $O(N log N)$.

2. Distributed N-body Simulation on the Grid

The main advantage of grid infrastructures from the users perspective is easy and flexible access to computational power. Therefore many computational science domains are increasingly relying on grid computing for running their simulations. However grid infrastructures are heterogenous and dynamic in terms of resource availability reducing their usability for parallel simulations, which are commonly designed to run on high-performance computers. Thus harnessing the computational power of grid infrastructures requires adapting the computational codes accordingly. This poses a specifically challenging task for N-body simulations, which exhibit highly irregular computation and communication patterns.

For investigating the distributed execution of a hierarchical N-body simulation on a heterogenous grid infrastructure a prototype implementation based on MPI and the pthreads-API has been developed. To enable the distributed execution of a single MPI application spanning several grid resources PACX-MPI [2] has been used while relying on the glogin tool [5] for starting up the distributed MPI application. For initial tests on the AustrianGrid infrastructure the distributed N-body code has been started on several grid resources using the glogin remote command invocation functionality. The MPI library is used as a transport layer for messages between grid resources as well as intra-resource communication for resources lacking a single system image such as clusters. Grid nodes offering shared memory multiprocessing can either use their local MPI implementation or they are handled by a single MPI process while relying on multithreading to harness the available processors.

3. The Distributed N-Body Tree

The hierarchical N-body application we have implemented as a prototype uses a binary tree with tight bounding boxes thus limiting the amount of interactions during force computation. The tree is built iteratively by subsequently adding the particles and continuously updating the tree data structure. Each new body is assigned to the nearest node, which is already in the tree. Since a leaf may only contain one particle a new inner node is also generated and inserted into the tree. The inner nodes of the tree maintain the bounding box, the mass center and the total mass for their subtree, while the leaves contain the same info for their particles. The tree data is stored in a sequential list containing the nodes in depth-first order. The N-body tree is maintained as a distributed global tree: Each processor holds a partial copy of the overall tree containing the minimal information required to handle the simulation of local particles, while subtrees stored on remote nodes are represented as proxy nodes, which contain the overall info for the remote subtree as well as the location of the process maintaining the data. If a specific node is represented within a local tree all its siblings are also available as proxies. Each tree needs to be represented starting from the root node to determine the position of the local particles within the global tree during the simulation. However if a process is not responsible for the

simulation of its upper nodes they are marked as invalid.

4. Tree Distribution

The heterogenity of grid infrastructures together with the irregular computation and communication patterns of a hierarchical N-body simulation turns the initial data distribution as well as dynamic load balancing into a crucial point for a viable grid-based simulation. Starting with an initial tree data structure located at a single grid resource and representing the complete input data, one can view the initial data distribution as a special case of dynamic load balancing.

Within our implementation the initial distribution as well as load balancing is adressed by identifying subtrees which need to be migrated to improve the load balance of the application. If a specific subtree is migrated away from a node it is extracted by copying it together with the path from the root node to the subnode to be extracted. This copy is subsequently sent to the recieving node, while within the local tree the node is turned into a proxy node and its subtree is removed. Upon receiving such a partial tree a node integrates the subtree into its own local tree.

5. Distributed Simulation

After the tree has been initially distributed the following steps are executed for each timestep of the simulation:

1. Computation of forces affecting particles
2. Calculating new particle positions
3. Updating the collective information to maintain consistency

As described above the force computation step is crucial due to its computational complexity. While the BH algorithm calculates all relevant force influences on a specific local particle it may require more detailed information for a specific remote subtree than being available locally. To limit the idle time during the simulation all remote tree parts possibly required for the force calculation are sent by their maintaining processes. Since every process can determine the area of influence of its local particles it can send the tree refinements without needing a specific request thus saving communication time [4]. After this communication step each process holds its locally essential tree (LET) [6] containing all required information for the force calculation on local particles.

After the forces have been accumulated using the BH algorithm, the additional refinements for the LETs are discarded and the new velocities and posititions of the particles are calculated using a forth-order Runge-Kutta integration method. This step can be performed without communication and has a linear complexity in terms of the number of particles.

During the third step of the simulation the collective information within the inner tree nodes is updated. At first every process sends out an update request for each of its proxy nodes. The reply contains the new collective information for the proxy node after the particle movements such as e.g. the updated bounding boxes. The update information received is propagated upwards within the local tree.

A specific problem for a distributed simulation is the synchronization which is required between the simulation phases. To provide an efficient and flexible synchronization mechanism enabling the usage of different termination conditions for different phases of the simulation a distributed termination algorithm [3] has been implemented.

6. Conclusions and Future Work

This paper describes our initial implementation of a distributed N-body simulation for a heterogenous grid infrastructure as well as the techniques applied to enable it. However the main obstacle for an efficient usage of grid infrastructures for distributed N-body simulations is given by the heterogenity of the communication and computation patterns of the application as well as the heterogenity of the underlying grid infrastructure. While the technical methods for adressing this heterogenity such as subtree migration for dynamic load balancing have been implemented, their application to support an efficient execution of the distributed application still needs to be investigated in more detail. In addition, the efficiciency of the parallel implementation is still being improved.

References

[1] J.E. Barnes, P. Hut, "A Hierarchical O(N log N) Force Calculation Algorithm", *Nature*, **324**, 4, 446-449, December 1986.

[2] K. Dichev, S. Stork, R. Keller, "MPI Support on the Grid", *Computing and Informatics*, **27**, 2, 213-222, 2008.

[3] E.W. Dijkstra, H. Feijen, A.J.M.V. Gasteren, "Derivation of a Termination Detection Algorithm for a Distributed Computation", *Information Processing Letters*, **16**, 5, 217-219, 1983.

[4] P. Liu, S.N. Bhatt, "Experiences with Parallel N-Body Simulation", *IEEE Transactions on Parallel and Distributed Systems*, **11**, 12, 1306-1323, December 2000.

[5] H. Rosmanith, D. Kranzlmüller, "glogin - A Multifunctional, Interactive Tunnel into the Grid", *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, November 2004.

[6] M.S. Warren, J.K. Salmon, "Astrophysical N-body Simulations Using Hierarchical Tree Data Structures", *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, 570-576, November 1992.

Gridification of the Solution of highdimensional improper Integration and Integral Equations

P. Zinterhof + P. Zinterhof jun. - Salzburg University

Abstract. *The scope of the paper is the development of new methods for high dimensional numerical integration of unbounded functions, the solution of integral equations and the direct search for good lattice points in a grid environment. From the mathematical point of view we use the Zinterhof sequences for the removal of singularities by means of certain integral transforms. The algorithms for numerical integration (by summation and convolution) and a brute force search for good integration numbers have been parallelized for use in the grid environment. We also show the high suitability of novel computing machinery, such as the STI Cell processor and CUDA-enabled devices (NVIDIA GPUs) for these tasks.*

1. The mathematical Methods

1.1. Numerical Integration Methods avoiding the Curse of Dimensionality

Consider any classical integration method

$$\sum_{n=1}^{N} W_n f(x_n) - \int_0^1 f(x)dx = R_N(f) \tag{1}$$

A straight forward generalization of (1) to s dimensions is the cartesian product rule

$$\sum_{n_1=1}^{N}\sum_{n_2=1}^{N}\cdots\sum_{n_s=1}^{N} W_{n_1}W_{n_2}\dots W_{n_s} f(x_{n_1}, x_{n_2}, \dots, x_{n_s}) - \int_0^1\cdots\int_0^1 f(x_1,\dots,x_s)dx_1\dots dx_s = R_N(f) \tag{2}$$

The computational time grows as N^s. If e. g. the computation of an onedimensional integral (1) using $N = 10^6$ nodes deserves one second, the computation of the correspondending 10-dimensional integral deserves 10^{60} seconds. This is called the "curse of dimensionality". The theory of uniform distribution of sequences provides best possible methods, essentially avoiding the curse of dimensionality.

We provide in this paper methods for the computation of improper integrals, where the singularities are on the boundary of the s-dimensional unit cube $E_s = [0,1]^s$. It is supposed, that the integrand $f(x_1,\dots,x_s)$ is smooth (differentiable) in $(0,1)^s$.

We propose in this summary some integration methods which are essentially best possible. A comprehensive expository source is [Drmota and Tichy, 1997]. The rather extended proofs will be presented in a forthcoming full paper.

Let $\{x\}$ be the fractional part of the real number x, $\{x\} = x - \lfloor x \rfloor$, $\lfloor x \rfloor$ being the integer part of x. We use the integration nodes (vectors $\in \mathbb{R}^s$).

$$\left(\{ke^1\}, \{ke^{1/2}\}, \ldots, \{ke^{1/s}\}\right), \; k = 0, \pm 1, \ldots, \pm(N-1) \tag{3}$$

The proof of the quality and efficiency of nodes of this type, which are frequently called Zinterhof sequences [Zinterhof, 1969], [Hofbauer et al., 2006], [Zinterhof, 2002], uses deep results of the Fields medallist Allan Baker e. g.

Let $s(x)$ be a growing and differentiable function, such that $s(0) = 0, \; s(1) = 1, \; s'(0) = s'(1) = 0$. The functions $s_1(x) = \frac{1}{2}(1 - \cos \pi x), \; s_1'(x) = \frac{\pi}{2}(\sin \pi x)$ and $s_2(x) = x - \frac{1}{2\pi} \sin \pi x, \; s_2'(x) = 2 \sin^2 \pi x$ are useful examples of such functions.

We use the weights $W_{N,k} = (N - |k|)/N^2, \; k = 0, \pm 1, \pm 2, \ldots, \pm(N-1)$ and consider the integration method

$$I_N(f) - I(f) = R_N(f), \tag{4}$$

$$I_N(f) = \sum_{k=-(N-1)}^{N-1} \frac{N - |k|}{N^2} f(s(\{ke^1\}), s(\{ke^{1/2}\}), \ldots, s(\{ke^{1/s}\})) \prod_{l=1}^{s} s'(\{ke^{1/l}\}). \tag{5}$$

If the function $f(x_1, \ldots, x_s), \; 0 < x_i < 1, \; i = 1, \ldots, s$ fulfills mild differentiability conditions, then the error term $R_N(f)$ of the integration process tends fast to zero:

$$R_N(f) = \mathcal{O}\left(\frac{1}{N^{\alpha - \varepsilon}}\right), \tag{6}$$

where α is depending on the smoothness of $f(x_1, \ldots, x_s)$ and arbitrary $\varepsilon > 0$. The result can not be better than $R_N = \mathcal{O}(\frac{1}{N^\alpha})$. So, the proposed method is essentially best possible. We propose the following set of test functions:

1.1.1 $f_1(x_1, \ldots, x_s) = (\max(x_1, \ldots, x_s))^\beta, \; \beta > -1$

with $I(f) = s/(s + \beta)$. We remark that the function $f_1(x_1, \ldots, x_s)$ is not separable, i. e. it is not a product of functions of fewer variables.

1.1.2 $f_2(x_1, \ldots, x_s) = (x_1 x_2 \ldots x_s)^{-0.5}, \; I(f_2) = 2^s$

1.1.3 $f_3(x_1, \ldots, x_s) = \prod_{l=1}^{s} (\ln(\frac{1}{x_l}))^{-0.5}, \; I(f_3) = \pi^{\frac{s}{2}}$

1.1.4 $f_4(x_1, \ldots, x_s) = \prod_{l=1}^{s} (1 - \frac{\pi^2}{6} + \frac{\pi^2}{2}(1 - 2\{x_l\})^2), \; I(f) = 1$

The function f_4 is the product of normalized Bernoulli polynomials of other two. It is essential for the definition of the so-called diaphony of a sequence. The diaphony measures the uniform distribution of a sequence and is a very useful estimator for the integration error.

The (Zinterhof-) nodes $(\{ke^1\}, \{ke^{1/2}\}, \ldots, \{ke^{1/s}\}),\ k = 0, \pm 1, \pm 2, \ldots$ have three very valuable advantages: the nodes provide fast integration methods. The previos nodes are not to be altered, if one increases the number N of nodes to be useful. And, last but not least, the nodes are readily computed for each N and every dimension s. On the other hand, the so-called good lattice points (i. g. optimal coefficients) or the so-called (t, m, s)-nets provide slightly smaller error terms, at least in theory. Unfortunately, both types of nodes deserve in general remarkable computing effort to be computed. So, there exist extensive tables of good lattice points and of (t, m, s)-nets, which are necessarily restricted to special N and s. The nodes used by us need no tabulation, because they are computed for any number N of nodes and any dimension. Nevertheless we present a grid version for brute force search for good lattice points.

1.2. Application of the proposed Integration Methods to Integral Equations Fredholm II

We consider the approximate solution of the Fredholm Integral Equation of second type in dimensions:

$$\varphi(P) = f(P) + \lambda \int_{G_s} K(P, Q)\varphi(Q)dQ, \tag{7}$$

where $G_s = [0, 1]^s$ and $\lambda \in \mathbb{C}$ is the parameter. The function $f(P)$ and the kernel $K(P, Q)$ are known. We choose nodes $Q_1, \ldots, Q_N$ and solve the linear system

$$\psi_k = f(Q_k) + \lambda \frac{1}{N} \sum_{n=1}^{N} K(Q_k, Q_n)\psi_n,\ k = 1, \ldots, N. \tag{8}$$

The solution is unique for small $|\lambda|$ at least. The approximate solution $\varphi_Q(P)$ is defined by

$$\varphi_Q(P) = f(P) + \lambda \frac{1}{N} \sum_{n=1}^{N} K(P, Q_n)\psi_n. \tag{9}$$

It is well known, that the error term $R_N = \varphi(P) - \varphi_Q(P)$ has the same order of magnitude as the error term in the corresponding integration process, where good estimations for the $\mathcal{O}$-constants are available. Nevertheless, in higher dimensions one needs many nodes $Q_1, \ldots, Q_N$ to achieve a reasonable small error term in (9). So, the solution of the linear system (8) is crucial, say for $N = 10^6$. We consider therefore two subtypes of Fredholm II, namely the convolution type and separable type

1.2.1 The convolution type

$$\varphi(P) = f(P) + \lambda K(P)\varphi(P) \tag{10}$$

In this case the linear system (8) is circulant and will be solved by three FFT. So the computing time reduces from $\mathcal{O}(N^3)$ to $\mathcal{O}(N \ln N)$.

1.2.2 The separable type

Here are kernels $K(P, Q) = \sum_{l=1}^{L} A_l(P)B_l(Q)$ considered, where $L \ll N$. The crucial point of

the numerical solution in that case is the numerical integration of the products $A_l(P)B_m(Q),\ l,m = 1,\ldots,L$, and the solution of a $L \times L$ linear system. The computational effort is in this case of order $N \times (L^2 + L^3)$, which is much less than N^3 in the general case.

1.3. Optimal nodes for integration

Let

$$H(z) = \frac{1}{p}\sum_{k=1}^{\frac{p-1}{2}} (1 - 2\{\frac{k}{p}\})^2(1 - 2\{\frac{kz}{p}\})^2(1 - 2\{\frac{kz^{s-1}}{p}\})^2 (11)$$

with $\{x\}$ =fracpart$(x), 0 \leq \{x\} < 1,\ \forall x$
now, find the minium of $H(z)$ within the range of integer values of z with $1 \leq z \leq \frac{p-1}{2}$. If this minimum is considered at $z = a$, $H(a) = \min_{1 \leq z \leq \frac{p-1}{2}} H(z)$, then the P vectors $(\{\frac{k}{p}\}, \{\frac{ka}{p}\}, \{\frac{ka^2}{p}\}, ..., \{\frac{ka^{s-1}}{p}\}),\ k = 1, .., P$ are optimal nodes for numerical integration and related problems.

2. Implementation for the grid

As shown above, numerical integration requires large amounts of floating-point operations. Furthermore, the quality of the results is proportional to the number of integration nodes, which is the motivation for the use of very high numbers of integration nodes. To make the computation of numerical integration more feasible for users without excessive hardware resources, we provide a framework of computer programs, which allow the computation of such tasks within a grid environment. Thus, the user can easily make use of the greater hardware resources offered by the grid, while being freed from the hassles of programming and tuning his algorithms for parallel execution. Another benefit of our implementation of numerical integration routines for the grid is the user's possibility to do more computations per time interval (e.g. repeat an integration for many different numerical settings) and to get better precision of these integrations, due to the possibly higher numbers of integration nodes. During the project we encountered many new hardware developments, such as the Sony/Toshiba/IBM Cell Broadband Engine Architecture (Cell processor) and NVIDIA'a general purpose graphics processing units (gpgpu, [NVIDIA, 2008]). Besides many other interesting accelerator plattforms (e.g. Convey's systems of tightly intergrated Xeon/fpgas and Sicortex systems), those two plattforms, while currently not being in the mainstream of typical grid hardware, look rather promising to us for two reasons: First, they are affordable, which makes them excellent candidates for building larger high performance cluster systems, that can be intergrated into grid environments. Second,they offer tremendous performance for single precision floating-point operations and very good performance on double precision codes, compared to the ubiquitous Intel x86-based systems. We therefore ported most of our algorithms to Cell and GPU platforms and show the beneficial role they can play in numerical integration tasks.

In the following we give an overview of the concepts of gridification and implementation details for the three main chapters (1.1 - 1.3). Each section is accompanied by practical results (e.g. precision) and benchmarks.

2.1. Numerical integration

The numerical integration of Fredholm type II integrals requires large summations of function evaluations. Due to the fact, that the error decreases in proportion to the number of integration nodes, we aim for very large - usually in the range of billions - integration node numbers. To achieve reasonable execution times we "gridified" the former sequential algorithm by dividing the summation into smaller blocks, that can be executed in parallel within the grid environment. During the test runs we encountered the - practically anticipated - effect of uneven load-balance. This effect results from different computational power, different hardware architectures or different utilization of the grid hardware and it leads to sub-optimal execution times, in that some grid nodes finish their blocks of work earlier than others and staying idle for the rest of the time. We therefore implemented a simple but rather effective scheme of automatic load-balancing, that doesn't require any previous knowledge of the current utilization of the grid nodes, at the expense of a slightly increased network traffic and very decent administrative overhead at the main node.

The code consists of an spmd-style (single program, multiple data) program, that is based on the MPICH2 (message passing interface) library. It therefore can easily be run on local clusters and grid-enabled machine sets, that support some form of MPI. The program master is started on one machine and replicates itself within the grid as $N-1$ worker nodes, resulting in N active processes. Process 0 (master) partitions the sum consisting of S sumands (e.g. $S = 1.000.000.000$) in k evenly sized blocks b_k (intervals) of size l, such that $S = kl$ (figure 1, upper row). Each worker then keeps asking for the next block to compute its local sum. When the worker has finished its local job of summing up the function evaluations within the given interval, the sum is returned to the master which adds it to the global sum. In order to reduce communications overhead between master and worker, the returning of some local sum triggers the master to send out the next interval to the very same worker node that has just finished. The underlying rationale of the scheme is to provide faster nodes with more work than slower workers. Without complicated monitoring this can be achieved by the described demand-driven method of load balancing.

To further improve performance we added support for CUDA devices (Compute Unified Device Architecture). Currently, NVIDIA is offering these devices either as high performance computing nodes (accelerator boards) or in the form of CUDA-enabled general purpose graphics processing units (gpgpu, or 'GPU' in short). Both types differ in main memory size and computing capability. In this paper we refer to GPUs. CUDA devices are massively parallel systems, which offer hundreds of tightly integrated processing elements within a large hierarchical memory system. They are programmed by way of the SIMD (single program, multiple data) method, which imposes certain limitations, such as the need for a great amount of data parallelism. Also, in order to effectively use the computational capabilities of GPUs, one has to employ thousands of parallel threads, so that the scheduler of the device can hide memory latencies and shortage of (fast) local memory. We chose a CUDA grid size of 256 blocks and 256 threads in order to supply enough tasks for the GPU hardware, which is capable of running more than 12000 tasks concurrently. For the most efficient usage of the GPU hardware, the size b_k of each distributed block within the mpi-layer has to be a multiple of $256 * 256 = 65536$. So the smallest possible number of integration nodes N on GPU type hardware would be $N = 65536$ (figure 1, bottom row).

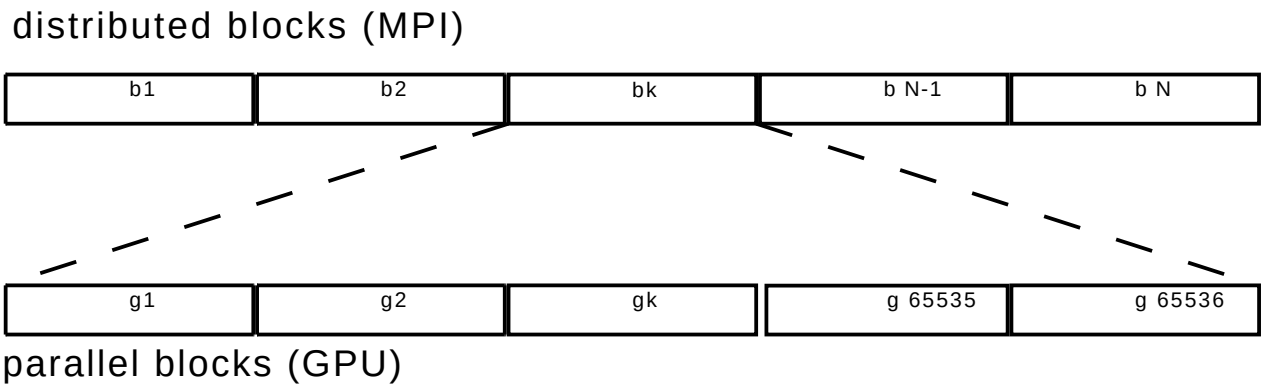


Figure 1. blocking scheme on global (MPI) and local (GPU) level

integration nodes	1 x86	4 x86	1 GPU (295)	4 GPU (295)	1 GPU (280)	error
65.536	0.15 s	0.5 s	4.65 s	5.05 s	5.6 s	$4.0 * 10^{-5}$
655.360	1.20 s	1.3 s	4.68 s	4.95 s	5.7 s	$1.02 * 10^{-6}$
8.388.608	19.13 s	5.1 s	4.89 s	5.07 s	6.04 s	$4.63 * 10^{-8}$
67.108.864	153.7 s s	39.7 s	7.05 s	5.76 s	8.54 s	$3.77 * 10^{-11}$
536.870.912	1240.5 s	315.1 s	22.59 s	10.07 s	28.71 s	$1.06 * 10^{-11}$
1.073.741.824	2455.7 s	622.8 s	40.42 s	14.34 s	51.72 s	$1.4 * 10^{-12}$
2.147.483.648	4925.0 s	1256.5 s	94.85 s	24.10 s	97.80 s	$9.1 * 10^{-13}$
4.294.967.296	9899.22 s	2467.2 s	147.19	41.84 s	158.80 s	$1.3 * 10^{-12}$

Table 1. execution times in seconds, example 1.1.2, s=5 (295=NVIDIA 295GTX, 280=NVIDIA 280GTX

The following performance results (table 1) were obtained on Intel Q6600 (2.4GHz) and NVIDIA GPU type machines. They indicate good scalability in the relevant range of node numbers ($N > 500.000.000$), which is key for the use within a dynamic grid environment. The sub-linear speedup for GPU-versions and low integration node numbers stems from CUDA-process startup times. For low integration node numbers the cpu versions show better (linear) scalability.

The mathematical precision of the integration is even better than expected. If we consider double precision numbers with a precision of 15 digits and the smallest possible error of 10^{-15} in each part of the computation, then by averaging out every second error term and a node number of approximately $2*10^9$ ($\approx 2^{31}$) we would expect an overall error of $2*\frac{10^9}{2}*10^{-15} = 10^{-6}$. Thus we would get about $15-5 = 10$ digits of precision, instead we achieve 12 digits of precision, which is two orders of magnitude better than expected.

2.2. Convolution type

The convolution type of numerical integration (10) is based on two large 1-dimensional discrete fourier transforms, which are inherently hard to parallelize on distributed memory systems. On typical grid sites we often encounter clusters of workstations, that are capable of massive numbers floating-point operations per second but suffer from rather weak system interconnects. So we didn't try to speed up the single FFT-operation but we built a framework to compute many such FFTs in parallel. This approach seams reasonable since many users might be interested in fast integration of many functions, rather than to solve a single integration in the

integration nodes	Intel Q6600 (2.4GHz)	280 GTX	speed up	error variance
1048576	2.58 s	0.617 s	4.18	$4.89*10^{-6}$
2097152	5.39 s	1.07 s	5.03	$1.31*10^{-6}$
4194304	11.51 s	2.03 s	5.66	$3.03*10^{-7}$
8388608	24.60 s	3.78 s	6.50	$1.26*10^{-7}$

Table 2. convolution (3x 1d-FFT): execution times in seconds

shortest possible time.

The performance values (table 2) were obtained for a 5-dimensional function on the Intel Q6600 (one core) and the NVIDIA 280 GTX GPU with 30 multiprocessors. The speedup calculation is based on the timing ratio (wall time)between the two compared systems.

We have to note that the GPU is only used for the FFT-part of the computations, which leaves the initial setup of the coefficient arrays to the cpu. In the final version of the code the setup process will be moved completely onto the GPU, which will further increase the speedup in favor of the GPU version.

2.3. Finding optimal nodes for numerical integration

As described in 1.3. optimal nodes for numerical integration can be found by a brute force computation of all $H(z)$ with $1 \leq z \leq \frac{p-1}{2}$. In other words: We have to compute a set of $\frac{p-1}{2}$ integrations and find the minimum value within this set. We chose the outer loop for parallelization, which reduces the communications overhead to $\frac{p}{2}$ task definition messages (master $\Rightarrow$ workers) and $\frac{p}{2}$ result messages (workers $\Rightarrow$ master). The algorithm has been implemented on standard x86 type cpu and the STI Cell processor, which is used in the SONY PS3 gaming console. The program for the Cell processor employs so-called spulets, which are stand-alone programs that can be executed directly on the synergistic processing elements and communicate with the calling code by way of a pipe. We now treat a single Cell system (PS3) as 6 separate processing elements, that is, from the MPI point of view, we launch 6 MPI-jobs on the ppu of every PS3. Each MPI-worker then acts as a gateway between the master node (by sending/receiving MPI-pakets) and one single spu-processor (Unix pipe). This approach enables seemless integration of the Cell's spu chips within our mpi-framework, including good load-balancing not only within the 6 on-chip spus, but also within distributed Cell chips.
Preliminary timing data show that the tremendous performance of GPUs can be fully utilizied for this brute force search algorithm, too.

Cell is especially well optimized for single precision calculations within streaming multimedia applications. Despite the fact that the brute force search requires double precision operations, Cell delivers deliberately higher performance than the Intel Xeon quad core chip that we also used for our benchmark runs. It seems especially noteworthy that a single PS3 gaming console with only 6 synergistic processing elements achieves a performance in the range of 2/3 of a dual quad-core Xeon server (table 3).

nodes p	1 Xeon 2.3 GHz	1 PS3 Cell	8x Xeon 2.3 GHz	8x PS3 Cell
10000	3.03 s	1.17s	1.03 s	2.97 s
20000	12.3 s	2.83 s	1.76 s	3.24 s
30000	27.3 s	5.21 s	3.90 s	3.56 s
40000	47.8 s	9.28 s	6.31 s	3.77 s
50000	74.0 s	13.66 s	9.75 s	4.29 s
60000	105.9 s	19.42 s	13.72 s	5.11 s
70000	143.4 s	26.80 s	18.62 s	5.88 s
80000	191.8 s	34.51 s	24.69 s	6.86 s
90000	235.5 s	43.18 s	30.03 s	8.07 s
100000	290.0 s	52.85 s	36.81 s	9.31 s
150000	647.9 s	117.78 s	81.42 s	17.47 s
200000	1147.7 s	207.18 s	144.34 s	28.62 s

Table 3. brute force search: execution times in seconds, dimension s=5

3. Conclusions

Regarding the application of our algorithms within the grid environment, we show good scalability and thus high suitability for the grid. Scalability is better for x86-type machines than for novel architectures such as the Cell chip and the GPU. This is a result of the more complicated startup routines, which include loading of drivers, calling of GPU-kernels and also from the rather low performance of Cell's ppu (power-pc unit). Nevertheless, the use of grid services along with novel computer architectures enables extremely high performance for the computation of numerical integrations and we believe that the integration of those architectures within the grid environment is a very promising step forward. Grid computing and the use of cutting-edge architectures significantly extends the area of practical computability of integration based numerical problems.

4. Acknowledgement

We want to thank G. Haase (University of Graz) for his support by giving access to his department's grid hardware as a very valuable extension of our local environment.

References

[Drmota and Tichy, 1997] Drmota, M. and Tichy, R. F. (1997). *Sequences, Discrepancies and Applications*. Springer.

[Hofbauer et al., 2006] Hofbauer, H., Uhl, A., and Zinterhof, P. (2006). A pragmatic view on numerical integration of unbounded functions. In Keller, A., Heinrich, S., and Niederreiter, H., editors, *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 512–528,. Springer.

[NVIDIA, 2008] NVIDIA (2008). *NVIDIA CUDA Compute Unified Device Architecture, Programming Guide*. NVIDIA Corporation.

[Zinterhof, 1969] Zinterhof, P. (1969). Einige zahlentheoretische methoden zur numerischen quadratur und interpolation. In *Sitzungsberichte der Österreichischen Akademie der Wissenschaften, math.-nat.wiss. Klase Abt. II*, pages 177:51–77.

[Zinterhof, 2002] Zinterhof, P. (2002). High dimensional improper integration procedures. In Technical University of Košice, C. E. F., editor, *Proceedings of Contributions of the 7th International Scientific Conference*, pages 109–115, Hroncova 5, 04001 Košice, SLOVAKIA. TULIP.

Prediction of Precipitation in the Alps using ASKALON on the Austrian Grid

Kassian Plankensteiner[†], Johannes Vergeiner[*], Radu Prodan[†], Georg Mayr[*] and Thomas Fahringer[†]

Abstract. *The ASKALON Grid Workflow System aims to help end-users in the development of scientific workflow applications for execution on Grid environments. In this work, we show how ASKALON was used to quickly adapt a meteorological application for numerical precipitation prediction, called LinMod, to be able to execute on the Austrian Grid infrastructure. We present the necessary steps that needed to be taken to make the existing application run as a parallel Grid workflow using the ASKALON workflow system and show performance evaluations based on actual executions in the Austrian Grid.*

1. Introduction

Numerical weather prediction (NWP) models are currently capable of predicting precipitation on the scale of 10 x 10 km. The LinMod application is down-scaling precipitation prediction to below 1km x 1km for 48 overlapping boxes covering the Alpine Region with a linear theory using influx conditions from a NWP and a finer-scale topography. Since weather forecasts are inherently uncertain due to the chaotic nature of the atmosphere, an important task in the down-scaling process is to investigate a range of parameters such as initial conditions, time-scales of conversion of cloud water to hydrometeors (rain or snow) to estimate the bandwidth of the resulting finer-scale precipitation patterns. Therefore, a sufficient prediction quality can only be achieved using a high number of model runs, each in itself only having small computational cost, making it an ideal match for Grid infrastructures.

The ASKALON Grid Workflow System[2] provides the means to reuse software components that were not specifically written for parallel or distributed infrastructures. ASKALON is able to make use of these legacy applications as building blocks for complex Grid workflow applications that can be executed on any modern Grid infrastructure. Additionally, it abstracts from the underlying infrastructure and its complexity and therefore provides the end user with an invisible Grid and a quick way to enable many applications to run as a parallel Grid workflow.

2. The LinMod Application

Primary precipitation forecasts are made with numerical weather prediction (NWP) models. Their spatial resolution is currently about 25 km or larger for global forecast models and between 2.2 and 14 km for regional forecast models. In mountainous areas like the Alps, the topography in the NWP

[*]Institute of Meteorology, University of Innsbruck, Austria
[†]Institute of Computer Science, University of Innsbruck, Austria, Email: kassian.plankensteiner@uibk.ac.at

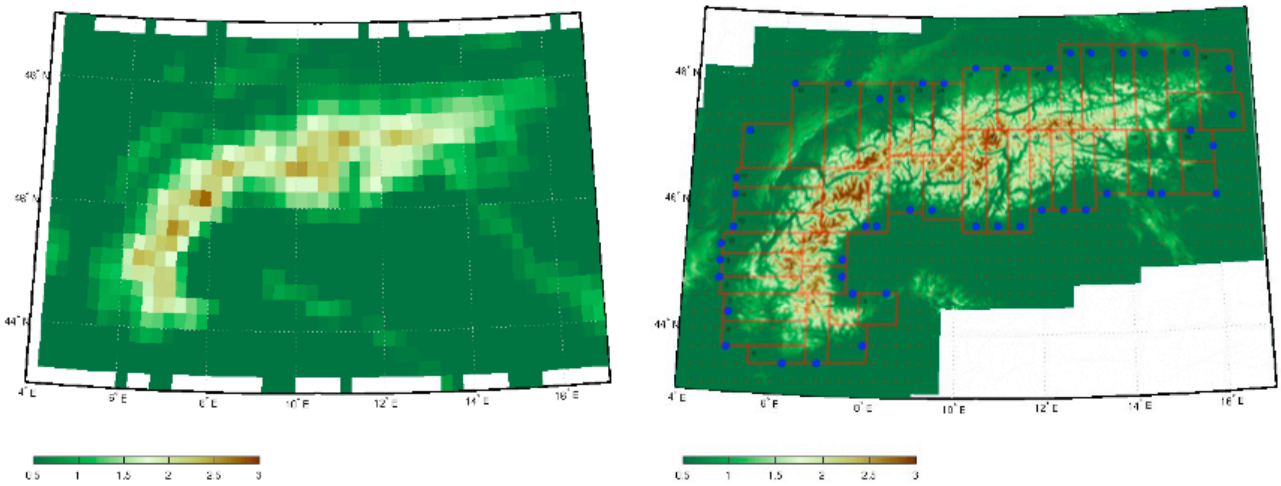


Figure 1. Comparison of Alps topography in a global Numerical Weather Prediction Model (ECMWF) on the left and the resolution used for the linear down-scaling on the right

model differs strongly from the real topography due to the coarse spatial resolution and the resulting smoothing of terrain features. Innsbruck, for example, is at 1100 to 1500 meters above mean sea level (MSL) depending on the NWP model, whereas its actual altitude is around 600 m MSL. Figure 1 shows a comparison of the topography data that is used in a global NWP model and the topography for the approach of linear down-scaling.

These differences between actual and model topography lead to large undesirable differences between predicted and observed precipitation patterns and amounts. Increasing the resolution of the general purpose models is limited by available computing capacity on one hand, and by having to adjust the model formulation and to add additional data sources for generating the initial conditions, on the other.

Since precipitation patterns and amounts in the Alps are largely controlled by the topography, the LinMod application uses a linear model [4, 5] to down-scale the prediction process to the desired resolution of about 1km x 1km. The benefit is the small computational cost for individual runs, which raises the opportunity to perform multiple runs to get a grip on the inherent uncertainty in the prediction. Two main approaches were followed here. First, instead of using input conditions (moisture influx, stability, wind) from one deterministic NWP run, we make use of the ensemble prediction system (EPS) of the European Center for Medium Range Weather Forecast (ECMWF). Second, the time constants for conversion of water clouds to hydro-meteors and for hydro-meteor fallout depend on several parameters (i.e. season, intensity of upward motion, topography, wind speed) and only the magnitude of their value is known. Therefore, a range from 500 seconds to 2500 seconds is covered in order to get a realistic bandwidth of the precipitation rate and spatial distribution.

For the down-scaling approach, the Alpine region is split into 48 separate boxes (see Figure 1) to get the local atmospheric conditions (e.g. wind direction, stability, moisture flux) right. Each box has a big overlap area (not shown in the figure) to avoid computational errors spreading from edges into the areas of interest.

All together, for one day of precipitation forecasts, the application has to execute a model run for 8 timesteps (three-hourly data) for all of the 48 boxes, each with 51 Ensemble Prediction System (EPS)

runs and 5 different time-scale values, leading to a total of 97920 independent model runs.

3. The ASKALON workflow system

ASKALON[2] is a Grid workflow system developed at the University of Innsbruck. Its main focus lies on enabling efficient development and execution of scientific workflow applications on distributed computing infrastructures, while shielding the end user from the complexities of the underlying systems. ASKALON consists of components responsible for workflow application development, execution, prediction, scheduling and monitoring as well as a software- and hardware-resource manager.

In ASKALON, Grid workflow applications are either specified in a UML-based graphical user interface or in the XML-based Abstract Grid Workflow Language AGWL[3]. Both of these specification methods support parallel loop constructs that can concisely express the large parallel sections of the LinMod application. ASKALON automatically translates this workflow specification to a concrete, executable representation at runtime.

3.1. The Porting Process

In the ASKALON system, a workflow consists of atomic and/or compound activities connected using data- and control-flow dependencies. Atomic activities have a certain activity type that is defined by its Input and Output Data Ports. Every Data Port has a value of a specific type, i.e. file, string, integer, double or sets of such typed values, which are called collections.

The identification of sequential and parallel sections in the application, which are then mapped to activities to be used in a workflow is an important step of the porting process. It is important to consider the fact that overheads in Grid environments can become rather large, especially for a high number of activites on strictly sequential paths. Therefore, one should carefully choose a grouping of adjacent sequential sections to single activities, to reach suitable granularity for execution in the Grid. This minimizes unnecessary overheads like scheduling, submission and queue wait times for the section because these overheads usually only occur once per activity in a workflow.

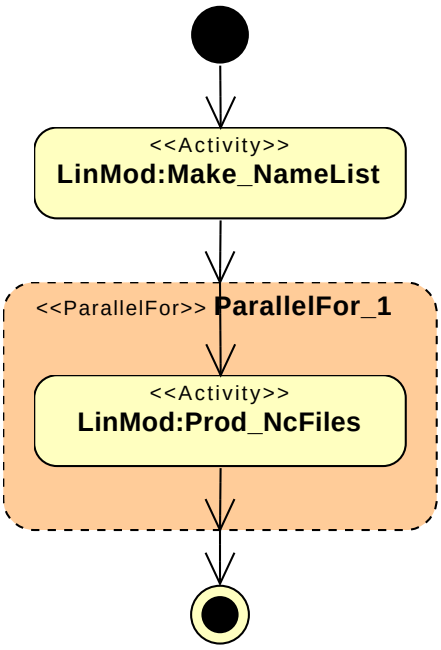


Figure 2. UML-based representation of the LinMod Grid Workflow

3.2. Creating Deployments

Deploying a new activity in a Grid to be utilized by ASKALON involves two distinctive steps. First, the user identifies the Data Ports of the activity and defines the activity type either by using a graphical user interface or by defining it in a Grid Workflow Deployment Definition (.gwdd) file. This definition includes the data port definition consisting of names and data types, an activity type name, a domain description as well as information about how the concrete deployments of the abstract activity type can be called and executed by the ASKALON system.

As a second step, the user installs the activity, potentially composed of several executables and libraries, all for example steered and executed via a single simple shell script, on every Grid site. This

can either be done manually, or by defining an automatic deployment specification and handing the task over to ASKALONs software resource manager, GLARE[1].

After specifying the activity types and deploying them across the Grid, they are registered in the GLARE database, thereby making them available to workflow application developers. The deploying user does not have to be the same as the user that develops the workflow itself. Deployment can be handled by system administrators of the corresponding machines or by people familiar with the applications to be deployed.

3.3. Creating the Workflow Application

Once the activity types are defined and deployed across the Grid, the activity types can be used as building blocks to create the workflow application. In the case of the LinMod application, Figure 2 shows the Grid workflow in the graphical UML-based representation that is used by ASKALON during the graphical workflow development phase.

The LinMod Grid workflow is composed of two top-level activities, *LinMod:Make_NameList*, an atomic activity and the composite parallel loop activity *ParallelFor_1*, which in turn is built up from one atomic activity per iteration, called *LinMod:Prod_NcFiles*. The first activity, LinMod:Make_NameList is responsible for the creation of the files that specify all of the parameters for the multitude of model executions carried out in the following ParallelFor loop.

This workflow is only specified in terms of activity types, which means that no information about the underlying execution infrastructure is shown to the developer of the workflow application. At runtime, the scheduler of ASKALON maps this abstract workflow definition to concrete activity deployments on the Grid that can be invoked by the execution service.

4. Experiments

We experimentally evaluate a first version of the Grid-enabled LinMod workflow application computing a deterministic run (without the Ensemble Prediction System) of the model. We show performance results as well as overhead and efficiency analysis based on executions on the Austrian Grid infrastructure. Furthermore, we present results of the precipitation forecasts computed by the workflow.

References

[1] Siddiqui et al. Glare: A grid activity registration, deployment and provisioning framework. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 52, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Thomas Fahringer et al. Askalon: A development and grid computing environment for scientific workflows. *Workflows for e-Science*, pages 450–471, 2007.

[3] Thomas Fahringer, Jun Qin, and Stefan Hainzer. Specification of Grid Workflow Applications with AGWL: An Abstract Grid Workflow Language. In *Proc. of IEEE CCGrid 2005*, Cardiff, UK, May 9-12, 2005. IEEE Computer Society Press.

[4] R Smith. A linear upslope-time-delay model for orographic precipitation. *Journal of hydrology*, Jan 2003.

[5] R Smith and I Barstad. A linear theory of orographic precipitation. *Journal of the Atmospheric Sciences*, Jan 2004.

Large scale data analysis on the grid in the search for new physics

D. Liko[*,†], W. Adam[*], B. Epp[‡], G. Mair[‡], N. Hörmann[*], G. Kasieczka[*], E. Kneringer[‡], P. Jussel[‡], P. Öttl[‡], H. Rohringer[*], R. Schöfbeck[*], J. Strauss[*], W. Waltenberger[*] and E. Widl[*]

Abstract. *The start-up of the LHC collider in autumn 2009 will allow particle physicists to explore a new territory at significant higher collision energy. This will not only allow to probe the current understanding of the nature of matter, but enable also the search for new physics beyond the standard model. At the same time also the challenges for data analysis are unprecedented: at the experimental site at CERN in Geneva data will be recorded in the PetaByte range. The AustrianGrid Service Centre with its federated Tier-2 for LHC will allow Austrian scientists to participate in the search for new physics in the first row. The recent increase in computing capacity in Austria allows for the first time to exercise the analysis chain on a realistic dimension. It was found that the specific requirements for data analysis have a strong impact on the operation. First experiences and further plans for operation and upgrades will be discussed.*

1. Introduction

After a long construction time the LHC project at CERN is entering this autumn a new phase: with the start-up of the collider physicists will finally be able to explore a new frontier in particle physics. The worldwide community of particle physicists has decided to combine their resources in this project to test the current understanding of matter at this crucial energy region: it is expected that the mechanism orchestrating the masses of particles and their interactions, the spontaneous breaking of the symmetry of the forces, is manifesting itself in that region as the notorious Higgs boson [1]. Of equal importance is the search for new effects, that would give insight into phenomena beyond the current understanding [2]. Austrian scientists are participating in this quest as members of the the CMS and the ATLAS experiment. Grid technology by itself and the AustrianGrid Service Center with its federated Tier-2 for LHC enables them to participate in the analysis of data at unprecedented scale.

The recent increase in computing capacity in Austria allows for the first time to exercise the analysis chain on a realistic dimension. In this presentation we review the operation of the experiments with a special emphasis on the tasks of the Austrian Tier-2. Some of the VO specific tools, which extend

[*]Institute for High Energy Physics, Austrian Academy of Sciences, Vienna, Austria.
[†]email: Dietrich.Liko@oeaw.ac.at
[‡]Institute for Astro and Particle Physics, University of Innsbruck, Innsbruck, Austria.

the more general middleware and which are in particular relevant for the operation in Austria, will be discussed. In preparation for the startup of the collider a number of exercises have been performed that should probe the scalability of the operation of the Tier-2 and demonstrate the status. While the experiences of these exercises have been satisfactory they have also demonstrated some limitations, which are being addressed with an ambitious work program.

2. Analysis of high energy physics data

The analysis of the data of HEP experiments requires a sophisticated chain of data processing and data reduction before they are examined by the physicist [3]. The processing chain will be performed on various computing centers in the grid. According to their location and capacity the centers have been arranged in a hierarchical system, Tier-0 being the computer center at the experimental site. 11 associated large Tier-1 centers take special roles in recording and the reprocession of data. The bulk of the data analysis will be performed on about 100 Tier-2 centers, as the one provided by AustianGrid.

In the first stage raw data is delivered by the data acquisition system of the experiment. The data consists out of the raw measurements of the detectors. These can be channel numbers, time measurements, charge depositions and other signals. Due to the large number of channels one can expect a single event to have a size in the order of several several megabytes. Integrated over the time of data taking during a year the overall data size is expected to be in the order of a PetaByte. For safty reason it is foreseen to store a copy immediately at CERN. After distribution of the data via the grid, a second permanet copy will be written at one of the large Tier-1 centers.

The data will then be processed in a complicated reconstruction step. For this step a precise knowledge of detector parameters is required. The specific geometrical coordinates of detector elements are typically refereed as the alignment and other relevant constants, as voltages and temperatures, as calibration. In practical terms it is required to determine this parameters by sophisticated alignment and calibration procedures before the actual reconstruction step. Using these parameters raw numbers are converted into physical parameters as space coordinates or energy deposits. A further pattern recognition step is translating these numbers in parameters of observed particles, characterizing them by their momentum and energy. This step will be typically performed at the experimental site, the Tier-0. At a later point the data might be reprocessed on the grid at the Tier-1 centers.

To simplify the analysis procedure only the most relevant quantities are stored in separate streams. These data is typically referred as AOD (Analysis Object Data). The size of the events is significantly reduced and the format provides the input for physics analysis. It is expected that the data for a year of data taking is in the order of 100 TB. This data has to be distributed to many sites on the grid and provides the input for a large number of analyses. As an example it is planned that the AustrianGrid Tier-2 for CMS will receive together with a small number of partner sites (Aachen, Bari, and Florida) a full set of data relevant for supersymmetry analysis. This does not only include the real event data, but also simulated events that are necessary to interpret the data. As data can be provided in several processing versions and is accumulated over the years careful planning will be required not to exceed the available storage capacity.

In the next step, the physicist will access the data and derive a further subset of the data, a so-called skim. The aim of this step is to reduce the amount of data to a reasonable value that can be hosted on a single site. This step is analysis dependent and can involve a further selection of interesting events. Often further data reduction is performed by calculating relevant quantities and dropping the more

detailed information of the previous processing step.

In a final step data will be analyzed by the physicists: Interactive tools are used to extract final quantities and visualize their behavior in plots and diagrams. These numbers are then input for discussions and finally publications of results. In the process of understanding the data it is clear that many of these steps have to performed in iterative fashion.

Details on the actual orchestration of the dataflow and the organization of the AustrainGrid federated Tier-2 for LHC are given in accompanying presentations.

3. Analysis Tools

Several middleware services had to be complemented by additional tools. There are two main areas: The first being the datatransport, that required additional services. It is discussed in more detail in a seperate presentation in this meeting. The second aspect are the computing services, that have required the development of specific analysis tools to support the users.

As an example the grid analysis tool used by the ATLAS collaboration, Ganga, is presented [4]. It has been developed with Austrian participation and is also used for applications outside high energy physics. Ganga is an easy-to-use frontend for job definition and management, implemented in Python. It has been developed to meet the needs of the ATLAS and LHCb for a Grid user interface, and includes built-in support for configuring and running applications for the two experiments. Ganga allows trivial switching between testing on a local batch system and large-scale processing on Grid resources.

A job in Ganga is constructed from a set of building blocks. All jobs must specify the software to be run (application) and the processing system (backend) to be used. Many jobs will specify an input dataset to be read and/or an output dataset to be produced. Optionally, a job may also define functions (splitters and mergers) for dividing a job into subjobs that can be processed in parallel, and for combining the resultant outputs. Ganga provides a framework for handling different types of application, backend, dataset, splitter and merger, implemented as plugin classes. Each of these has its own schema, which places in evidence the configurable properties. For the need of ATLAS a specific plugin has been developed that supports the ATLAS analysis framework Athena. The tool can export the development environment of a physicist and perform the processing of events at the remote site. It is fully integrated in the ATLAS data management system DQ2 and can automatically locate data on the grid sites. Built in support for job splitting allows to precess large dataset in parallel. Resulting datasets can be automatically registered and are available for further analysis. As of today GANGA has been used by about 2500 users in and outside high energy physics. On a day by day basis it is used by about 500 distinct users each month.

A similar tool has been developed in CMS, called CRAB. These tools provide the access to the grid to large user commonities. While some simplification in the use of grid resources have been optained, the actual operation requires a substantial support effort. The associated mailing list are amoung the most busy in the experiment. The reasons for this effort can be found in the various issues that are observed in an infrastructure of that size. At this point the users are not shielded from failuers and problems and have to take the required corrective actions.

4. Scalability test

The groups in Vienna and Innsbruck are exercising the machinery and are preparing their tools and programs for the startup. Exercises based on simulated events are performed to fine tune the selection mechanism that will extract the signals for new physics, when the data will be available in sufficient statistic. Results of such sample analysis will be presented.

At the same we are testing the grid tools and our infrastructure. We are identifying possible shortcomings. Software problems are address in close collaboration with the international development teams supporting the tools. In particular during the last month a general scaling test, STEP09, has been performed, that probes also the analysis capacity of the sites. Infrastucture problems are discussed within the federated Tier-2 center. Some examples of recent issues will be discussed.

The cluster in Vienna is also performing an ongoing capacity upgrade. This upgrade is driven by the requirements of data analysis on large scale and concerns the network and the capacity of the middleware services (Storage Element and Computing Element). It is planned that the final capacity will be reached with the start of the LHC operation.

5. Conclusions

In summary we can conclude that we enter well prepared to the startup of the collider. The Austrian-Grid Service Center with the federated Tier-2 for LHC is providing the infrastructure, which allows us to participate successfully to the analysis of the LHC data. It is well integrated into the global structure of the grid and provides Austrian scientics with the access to the worldwide resources.

Nevertheless we are also aware that the operation of the LHC collider will present us with a significant challenge also in the area of computing, both on the global scale and also in Austria. An obgoing upgarde activity is scaling the resources to the required level. As in physics we are probing a new territory.

References

[1] T. Wyatt, High energy colliders and the rise of the standard model, *Nature*, 448:274280, 2007.

[2] J. Ellis, Beyond the standard model with the LHC, *Nature*, 448:297301, 2007.

[3] D. Liko and N. Neufeld, Managing the extreme dataflow of LHC experiments, in it Encycopedia of Applied High Energy and Particle Physics, Wiley VCH Verlag GmbH, 2009.

[4] J.T. Mosciki et al, Ganga: A tool for computational-task management and easy access to Grid resources, *Computer Physics Communication*, 2009, doi:10.1016/j.cpc.2009.06.016.

Austrian Federated WLCG Tier-2

Peter Oettl*, Gregor Mair*, Katharina Nimeth*, Wolfgang Jais*, Reinhard Bischof†, Dietrich Liko‡, Gerhard Walzel‡ and Natascha Hoermann‡

Abstract. *The LHC (Large Hadron Collider) at CERN[1], the European laboratory for particle physics near Geneva, plans to resume operation and start data acquisition in fall 2009. The High Energy Physics Group at the Austrian Academy of Science and the Astro- and Particle Physics Group at the University of Innsbruck are part of the CMS (Compact Muon Solenoid) and the ATLAS (A Toroidal LHC ApparatuS) experiments. Both experiments have a high demand on computing power and data storage. To allow Austrian scientists to analyze the data the Ministry of Science and Research has provided funds for Grid computing for LHC in the context of the Austrian Grid project. It has also joined the Worldwide LHC Computing Grid Collaboration[2] (WLCG) by signing the Computing Memorandum of Understanding (C-MoU) [1]. To conform to this C-MoU a federated Tier-2 computing center has been installed.*

In this paper we introduce the structure of the WLCG, the resources required for the LHC experiments in general and for the two experiments CMS and ATLAS in detail. We discuss the setup of the Austrian Federated Tier-2 center, problems we faced, lessons learned and our future plans.

1. Introduction

The LHC plans to resume operation in fall 2009. The LHC experiments[3] - scientists in Austria are participating in the two large multi-purpose experiments ATLAS and CMS - will start collecting the long awaited data from the beginning. In total these experiments will produce up to 15 Petabytes of data per year. Over 5000 physicists at more than 500 research institutions and universities are interested in this data. To be able to process and distribute this amount of data the WLCG was proposed by CERN in 2001. Furthermore CERN is hosting the large EGEE[4] (Enabling Grids for E-sciencE) project funded by the European Commission, with the mission to deliver a sustained Grid infrastructure for a broad range of sciences. In close interaction with this project, Grid middleware has been developed. To validate the software and computing model Data Challenges have been performed with Austrian participation.

*Institute of Astro- & Particle Physics, University of Innsbruck, Austria
†ZID, University of Innsbruck, Austria
‡Institute of High Energy Physics, Austrian Academy of Sciences, Austria
[1]CERN, http://www.cern.ch/
[2]WLCG, http://www.cern.ch/lcg/
[3]LHC experiments, https://public.web.cern.ch/public/en/LHC/LHCExperiments-en.html
[4]EGEE, http://www.eu-egee.org/

The Astro- and Particle Physics Institute in Innsbruck set up their first WLCG Grid site in 2003 and participated in ATLAS Data Challenge 2[5] and large scale production for the physics workshop in Rome 2005. Since that time the site has been extended with computing and storage resources. The High Energy Physics Group in Vienna started to set up their first WLCG Grid site in 2005. Hence both institutes were able to gather their first experiences with WLCG. It soon became clear that Austria, as one of the smaller member states of CERN, has to combine its resources to form a federated Tier-2 center for ATLAS and CMS. The federated nature of this center allows it to profit on one hand from the common experience and on the other hand allows the local groups to concentrate on the specific work of their experiment. Therefore the Austrian Ministry of Science decided to fund a federated Tier-2 computing center and sign the C-MoU to be a collaborator of the WLCG. The resources of the Tier-2 allow Austrian researchers to access computing and storage resources of the WLCG and to perform their analysis.

In the next section we explain the structure of the WLCG, the computing and storage resources required for the LHC experiments in general and for CMS and ATLAS in detail. In section 3 we show the requirements to the Austrian Federated Tier-2 and its current status. We illustrate the setup of the Austrian Federated Tier-2, identified problems and how these were addressed. In section 4 we give an overview of recent tests and discuss the results and lessons learned. In section 5 we provide a conclusion and an outlook.

2. Worldwide LHC Computing Grid Structure

According to the LHC Computing Grid Design Report [2] the required CPU and storage capacities for all LHC experiments sum up to 140 million SPECint2k[6], to 60 PB of disk storage and 50 PB of mass storage in the first year of operation. The data will be distributed around the world according to a hierarchical tier based model. Raw data from the experiments is recorded on tape and initially processed at the Tier-0 centre at CERN. Then the data is distributed to 11 Tier-1 centers, which have sufficient storage capacity for a large fraction of the data and 24/7 operation. Tier-1 centers are also responsible for analysis tasks which require access to large subsets of data. The Tier-2 centers receive the data from their Tier-1 center. Tier-2 centers provide sufficient data storage and computing resources for user analysis tasks and Monte Carlo simulations. Through Tier-3 computing resources individual scientists and institutes may gain access to the Grid.

The WLCG is based on several Grid infrastructures: EGEE in Europe, OSG (Open Science Grid) in the US and the Nordic Data Grid Facility in the Scandinavian region. While these infrastructures support different middleware flavors, key components such as security, accounting and file transfer services are fully interoperable.

In Europe WLCG is built on the gLite[7] middleware provided by the EGEE project: the middleware includes components from Globus[8], Condor[9], the Virtual Data Toolkit[10] and gLite specific developments. On top of the middleware the experiments have developed specific Grid services that allow them to operate the infrastructure.

[5] ATLAS Data Challenges, http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/DC/
[6] SPECint2000, http://www.spec.org/cpu2000/results/
[7] gLite Lightweight Middleware for Grid Computing, http://glite.web.cern.ch/glite/
[8] The Globus Alliance, http://www.globus.org/
[9] Condor High Throughput Computing, http://www.cs.wisc.edu/condor/
[10] VDT Virtual Data Toolkit, http://vdt.cs.wisc.edu/

For data management the CMS experiment has developed the PhEDEx[11] (Physics Experiment Data Export) system. It provides a layer on top of the data services of gLite, the File Transport Service (FTS). This layer complements the services provided by the infrastructure and connects to the experiment specific bookkeeping system. The development of such an experiment specific layer also eases the interaction between sites, which might support different middleware flavors. The ATLAS experiment has developed a similar data transfer system called Don Quijote 2 (DQ2).

For job management PanDA[12] (Production ANd Distributed Analysis) is used in ATLAS. It was originally developed for US based production and has been in operation since 2005. It meets the requirements for a data-driven workload management system for production and distributed analysis capable of operating at LHC data processing scale. It has been adopted for production for all Grid flavors supported by ATLAS.

The collaboration members provide and operate a majority of the computing resources as part of the EGEE Grid, a consortium of 267 computing centers from 54 countries. These worldwide resources are complemented by resources in the United States being part of OSG and other resources in the Scandinavian area as part of the Nordic Data Grid Facility.

3. Austrian Federated Tier-2 Setup

The Austrian Federated Tier-2 supports the two LHC experiments CMS and ATLAS. The Virtual Organization (VO) CMS is supported at the resource center in Vienna and the VO ATLAS in Innsbruck. Both sites are used for production and analysis on a day by day basis. They also provide Austrian scientists access to the WLCG resources and allow them to participate in their analysis. In addition spare resources can be accessed via the VOCE[13] (VO for Central Europe) VO, a catch-all VO for users from the Central European federation.

Currently Innsbruck is operating two Computing Elements (CE). One managing 28 Worker Nodes (WN's) with a total of 224 cores and 2 GB memory per core. The other managing 9 WN's with a total of 36 cores and 2 GB memory per core. The 49 TByte disk storage in Innsbruck is managed by DPM[14] (Disk Pool Manager). Vienna is currently operating 416 cores and 100 TByte of disk storage managed by DPM as well. For Vienna it is foreseen to upgrade to around 1000 cores and 500 TByte of disk storage within 2009.

Austria's pledges for the Austrian federated Tier-2 according to the C-MoU are shown in table 1.

Table 1. Pledges for Austrian Federated Tier-2 according to MoU[1]

	2008	2009	2010	2011	2012	2013
CPU (kSI2k)	540	1060	1200	1300	1300	1300
Disk (TB)	110	300	330	390	440	440
Nominal WAN (Mbit / s)	1000	1000	1000	1000	1000	1000

Table 2 shows the resources currently provided.

[11] PhEDEx, http://cmsdoc.cern.ch/cms/aprom/phedex/
[12] PanDA, https://twiki.cern.ch/twiki/bin/view/Atlas/Panda/
[13] VOCE, http://egee.cesnet.cz/en/voce/
[14] DPM, http://www.gridpp.ac.uk/wiki/Disk_Pool_Manager

Table 2. Resources provided by Austrian Federated Tier-2

	Vienna	Innsbruck	Sum
CPU (kSI2k)	775	550	1297
Disk (TB)	150	49	199

Austria nearly fulfills the CPU pledges for 2011 and will be able to fulfill the disk pledges for 2009 with the next upgrade of storage.

In the presentation the setup of the two sites will be presented in more details.

4. Recent Tests

During spring 2009 the exercise STEP09[15] (Scale Testing for the Experiment Program 2009) took place: the four large LHC experiments simulated data production at nominal rate. The data was distributed to and between the Tier-1 centers for processing. At the same time Monte Carlo simulations and user analysis were performed on the Tier-2 centers with the predicted nominal rates. User analysis on Tier-2 centers within EGEE was done through both WMS jobs submission and through pilot jobs. This exercise was intended as a last verification of the scalability and reliability of the WLCG infrastructure before the actual start of the LHC operation. We will report on the results and lessons learned from this exercise.

5. Conclusion and Outlook

In this section we will discuss proposed solutions to problems that were identified during STEP09 and other tests, furthermore we will present our future plans for the Austrian Federated Tier-2.

References

[1] Worldwide LHC Computing Grid Collaboration, "Memorandum of Understanding", *(CERN-C-RRB-2005-01/Rev.)*, 2009.

[2] The LCG Editorial Board, "LHC Computing Grid Design Report", *LCG-TDR-001 (CERN-LHCC-2005-024)*, 2005.

[3] The Atlas Computing Group, "ATLAS Computing Technical Design Report", *ATLAS-TDR-017 (CERN-LHCC-2005-022)*, 2005.

[15] STEP09, https://twiki.cern.ch/twiki/bin/view/LCG/WLCGStep09

Network Based Execution of Dynamic Workflows in Grid and Cloud Based Environments

G. Stuermer*, J. Mangler†, E. Schikuta‡

Abstract. *Workflow engines often being based on WS-BPEL, currently rely on a mix of recovery / modification strategies [1] that are either part of the workflow description, part of the workflow engine, or realized as plugins to the workflow engine. To foster the development of distributed cloud-based workflow engines and novel repair algorithms, workflow engines have to be modularized in order to overcome the static and inflexible APIs provided by these workflow engines. Dynamic features gained by modularization include the creation of external modules to monitor as well as modify workflows to provide for error handling in combination with Service Level Agreement (SLA) constraints. This paper presents the architecture of a flexible Workflow Execution Engine that utilizes cloud based services for tasks like monitoring or repair and is able to move the workflow enactment itself to the cloud.*

1. Introduction

Cloud environments foster the "Software as a Service" [2] paradigm, in providing easy access to replaceable pieces of infrastructure, like storage, CPU power or services implementing arbitrary functionality. When using these pieces of infrastructure in a business environment, they become part of workflows, which are handled by workflow engines [3, 4, 5]. Workflow engines are specialized in orchestrating business processes consisting of multiple activities, these activities are commonly realized as web services. We believe that a cloud is the perfect execution environment for this kind of activities. To justify this statement we developed a modular workflow execution engine called WEE.

Definition of Terms. Before continuing we have to define some essential terms common in literature. We define a workflow as a behavioral description, that combines several external services (i.e. web services described by a WSDL). A workflow description consists of a set of control structures and activities that call the above mentioned web services. When such a description is executed it results in a workflow instance that holds the runtime information and associated data which is further referred to as workflow context. We further define the set of control structures and activities in a running workflow instance as workflow structure. Activities are defined as the sum of information associated with the execution of an external service (e.g. state, return values), provided by a cloud. For a running workflow instance, the thread of control is a sequence of activities constrained by the control structures. A workflow engines copes with all different aspects of workflow enactment, control flow, data, exception handling, resources, logging, security and others. A workflow execution engine is a part of a workflow engine that only deals with the control flow aspect including the invocation of

*Faculty of Computer Science, University of Vienna, Austria, email: a0302949@unet.univie.ac.at
†Faculty of Computer Science, University of Vienna, Austria, email: juergen.mangler@univie.ac.at
‡Faculty of Computer Science, University of Vienna, Austria, email: erich.schikuta@univie.ac.at

external services.

Moving the workflow engine to the cloud. In this paper we explain how to move the workflow engine itself to the cloud by modularizing the workflow execution part up to a point, where it can handle the requirements of dynamic workflows [6]. Dynamic workflows provide the possibility to intervene into the control flow and the context of a workflow at runtime. This is important for cloud based environments, because:

- In contrast to traditional workflow engines, there is no control over the availability of services, so they have to be replaceable at runtime.
- Integrating services from external vendors is different from orchestrating a workflow in that important information about an activity is hidden inside the external service. In case of a problem it is desirable to involve the external service in the repair process, because it has access to problem specific information.

Modularizing and moving the workflow engine to the cloud has the goal to standardize ways of interaction between the workflow engine and cloud services. Modularizing the workflow engine helps to clarify security aspects. Security in cloud based systems basically means trusting service providers [7]. Having separated contexts with well defined ways to access/modify information will provide starting points for security mechanisms.

Design rationale for a modularized workflow engine. We envision workflow execution as the interaction of highly independent modules, each of them covering a certain execution aspect. In order to create a workflow execution engine that enables to carry out a dynamic workflow execution we specify the following design rationale:

- The core of such an interaction is the workflow execution engine.
- The execution engine features a minimal set of operations to cover custom control flow patterns.
- The execution engine is not responsible for handling interactions with external services, monitoring, security, and repair strategies other than exceptions which are part of the workflow description. Each of these responsibilities is delegated to arbitrary external modules through a unified interface. These external modules are further called handlers.
- The execution engine is able to be started, stopped and continued.
- While the execution engine is stopped
  - the thread of control is modifiable,
  - the workflow structure is freely changeable (add/replace/remove activities, change control structures), and
  - the workflow context is modifiable (e.g. to fit a different thread of control).
- It is possible to replace handlers on the fly.

- The handlers themselves are able to trigger all functionalities mentioned above, and therefore are able to intervene with the control flow via the execution engine.

The flexibility gained through realizing the requirements allows to create a loosely coupled system consisting of the workflow execution engine and the handlers, that enables to use runtime-information to dynamically change the workflow in the above mentioned manners.

Structure of the paper. In Section 2. we will further elaborate on the desired properties of dynamic workflows, analyze its possible applications and benefits and compare it to state of the art approaches. In Section 3. explains how a modularized workflow execution in conjunction with the cloud can lead to novel business concepts, which is the justification for designing the WEE. In Section 4. we introduce the modular architecture of the WEE also by elaborating a small example. In Section 5. we explain the different types of handlers that are possible within our architecture and the interfaces they use to communicate with the WEE. In Section 6. we will conclude with an outlook on future research to employ the characteristics of our newly developed workflow execution engine.

2. The Concept of Dynamic Workflows

We define dynamic workflows as the possibility to intervene into the control flow and the context of a workflow at runtime. This includes minor changes like modifying an endpoint or context variable but also the replacement of a whole workflow.

To explain the benefits of dynamic workflows, we have to start with the following problem definition:

Diluted Business Code. When designing a workflow, the designer has to consider a multitude of possible errors. For example when services become unavailable a valid compensation strategy has to be in place. The multitude of error handling additions has to be seen as technical or administrative overhead, which dilutes the original aim of the workflow [8] and makes it more difficult to read and maintain the workflow description in respect to the underlying business process. Therefore, it has to be possible to handle certain errors by defining independent and separated strategies and correct them when they occur, e.g. by modifying the endpoint and redoing the execution.

Preemptive Problem Handling. A workflow execution engine has to pro-actively react on problems before they escalate. Possible applications include temporal conformance checking [9] and other issues related to Quality of Service (QoS) or Service Level Agreements (SLAs) [10]. An external monitor should be responsible for checking the compliance of certain constraints. Whenever it detects possible problems, it may decide to choose new services and even restructure the original workflow.

Builtin Repair Strategies. Existing Workflow Engines define their own repair strategies or at least provide a small set of possible options to carry out repair actions [11] (see efforts taken in [1] or [12]). These strategies are built directly into the workflow engines. They can be neither changed nor is it easy to include repair strategies according to events at runtime. Dynamic workflows have not only to allow for minor changes to the context but also for injecting compensating activities or even the complete replacement of the running workflow. However, it is not the responsibility of the execution engine to provide predefined ways to repair a failed workflow. A repair strategy should be adoptable to the reason of failure and may vary from case to case even within one flow of execution.

As a consequence of the above mentioned issues it should be the sole responsibility of an execution

engine to provide flexible mechanisms for external modules to intervene into the execution. Repair strategies themselves should be better realized as external modules and associated with given workflows at runtime by providing proper access to runtime information of the execution engine.

The outcome of dynamic repair strategies cannot be predicted at design time which is one of the basic issues leading to dynamic workflows. This is due to the situation that the requirements are not fully available at design-time or the situation can change in a way not foreseen at design-time. So it is necessary to allow modifications which move the already running workflow back into a clean state. The decision how a broken workflow has to be repaired or enhanced cannot be made in advance. Therefore it has to be either made by a human or a specialized program.

As stated by Rinderle et al. [13] and others, interventions can occur either at the level of workflow description (which is the focus of research activities in the field of workflow adaptation) or at the instance level [14, 15]. The interventions covered by a workflow execution engine solely affect the *current instance* of the workflow. Therefore the workflow description is still valid and the base for subsequent instances of this workflow. The modifications cannot be applied to other instances, because they depend on the runtime attributes of the workflow instance. So when dealing with the three fundamental issues of workflow changes as summarized by Rinderle et al. [13] which are completeness, correctness, and change realization, only completeness is relevant for our approach. As we propose a workflow execution engine, completeness has to be validated by the controlling external modules, whereas the execution engine checks syntactic correctness only.

3. Workflow Execution in a Cloud Environment

In a cloud based environment where many providers of services exist it is a necessity that a workflow engine has to handle different tasks in a dynamic manner as compared to traditional business or scientific scenarios that workflow engines covered so far.

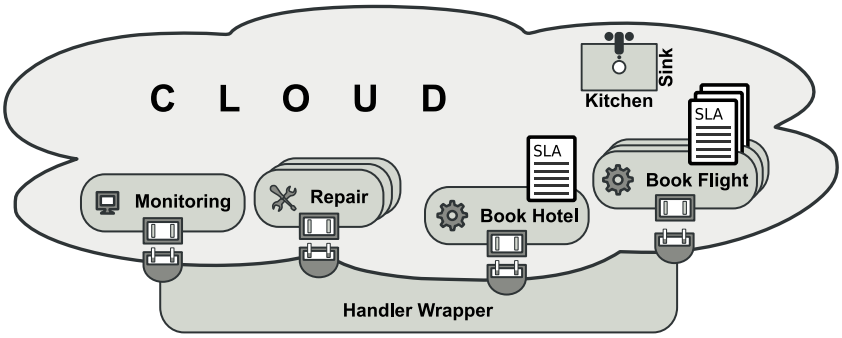


Figure 1. Workflow Execution in a Cloud Environment

With the ability to have external handlers for monitoring, repair and other tasks, several interesting topics come up. **Our vision** is to not only supply services to customers, but also repair handlers and monitors. These handlers would have the potential to react much quicker and much more precise on upcoming issues as they are not coupled with the workflow exposed by the customer but also know about the otherwise hidden context of the supplier. Also of interest can be custom repair algorithms, applied in conjunction with runtime information rather than specified at design time. Figure 1 shows how multiple services by multiple vendors could be used and tied together as handlers, included in a workflow through the handler wrapper.

A customer provides the workflow description that has to be executed by an execution engine. The execution engine itself may choose services and handlers according to the needs of the customer, specified by SLAs.

Also of interest for cloud based environments will be the security aspect. Although a handler provided by a supplier can use information from the supplier without exposing information to the outside world, interfering with the workflow requires a certain level of trust from the consumers side, which can be addressed by SLAs.

4. A Novel Architecture for Dynamic Workflows

In this section we illustrate the requirements of handlers and introduce different categories. We also show how a network of different handlers can be integrated into a workflow execution engine to foster the modularization of workflow execution.

Fig. 2 shows the architecture of the prototype implementation of the modular Workflow Execution Engine (WEE). The WEE is provided with a workflow description that has to be enacted. When the workflow is executed, the workflow environment contains additional information for the context (e.g. variables), the current state of execution (e.g. normal, failed, aborted) and the defined endpoints.

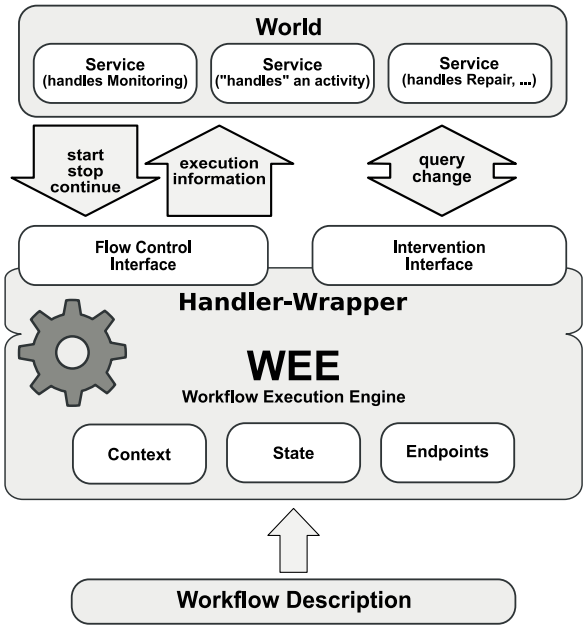


Figure 2. The WEE Architecture

WEEs sole purpose is to take care of the control flow, it instantiates a given workflow description, but it *DOES NOT*:

- Execute web services defined in the workflow description.
- Log the thread of control.
- Check in any way the semantic correctness of the enacted workflow description.
- Manage any resources used during the workflow enactment.

What it *DOES* is including a handler wrapper that in turn delegates the tasks to services, which we further refer to as handlers. Therefore, a handler is a kind of a servant to the execution engine which runs the parts of workflow that are not in the core competence of the execution engine.

But the relationship between handler and execution engine can be twofold:

Supervision & Intervention. A handler can act as a supervisor of the execution engine, getting informed about the process of execution, or intervene in cases when repair is needed (e.g. when SLAs are violated). In this case the handler has to adhere to a protocol defined by the

handler wrapper. This protocol is inherent to the handler wrapper and not part of the workflow description.

Execution. The workflow description holds information how to call external applications / services. This information includes location, operation name and parameters. The handler wrapper invokes the service and hands the results back to control flow. How to invoke the service (whether it is a SOAP service, a Java class or a Phone call) is inherently implemented by the handler wrapper, the invoked service itself does not need to know anything. In contrary to *Supervision & Intervention* the protocol how to interact with the service, what to make of the return value, is the workflow description.

We want to explicitly point out that the above made distinction is NOT a way to differentiate between handler that monitor or repair the workflow, and services that are invoked as activities as part of the workflow description. They are both called handlers because they handle specific task in the context of the workflow, and they are both connected through the wrapper handler with the workflow.

The functionality of our architecture is also depicted in Fig. 3, by introducing a simple flight/hotel booking example. The WEE consists of 2 parts, the workflow and the handler wrapper. Handlers are services provided for example by the cloud.

Initially the workflow consists of two activities: book a flight and book a hotel. When the first activity is carried out the WEE hands over the information about the first activity to the handler wrapper which in turn calls the booking routine on the Aeroflot homepage AND informs the monitor about the progress. The result of the call to the Aeroflot homepage is then handed back through the handler wrapper to the workflow.

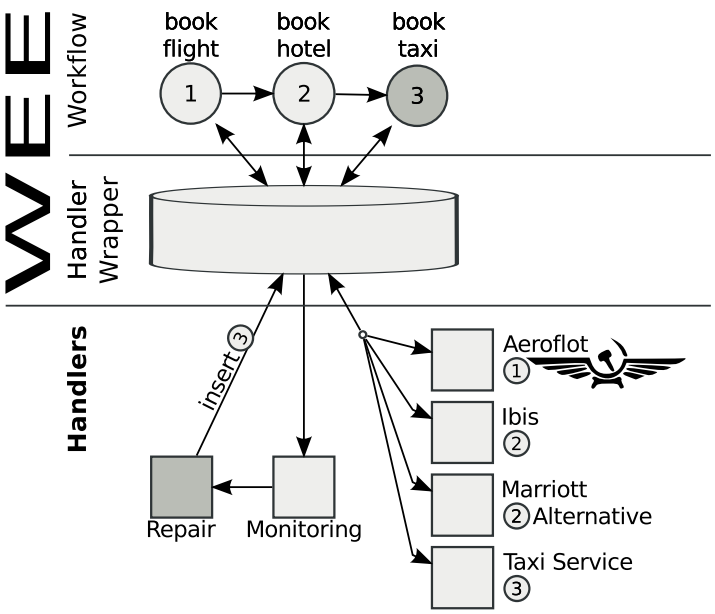


Figure 3. A Simple Flight / Hotel Booking Example

For the next activity the WEE again gives information to the Handler Wrapper, which first tries to book a room in the Ibis hotel. Because the hotel is already booked out, alternatively the room is booked in the Marriott hotel. Again the monitor is informed.

The monitor checks the situation and finds out that while for the Ibis hotel it is possible to walk from the airport to the hotel, for the Marriott one needs a taxi. Therefore it informs the Repair handler that inserts a third activity into the workflow, which is subsequently carried out.

5. Interfaces to Interact With the WEE

As depicted in Fig. 2, a handler can interact with the WEE through two interfaces:

The Flow Control Interface. On the one hand it provides means to get information about the process of execution. The information about processed tasks or exceptions are forwarded to the handler. In addition, the handler can start, stop and continue a suspended workflow execution. On the other hand it executes external services (or sends execution information to the monitor). Therefore the flow control interface manages *Execution* as mentioned above.

The Intervention Interface allows for enhanced manipulation functionality to address the specific needs of Dynamic Workflows raised in Section 2. These functions are:

Restructuring and Thread-of-Control Modification For Repair Purposes. This intervention has the most important effect on a workflow. Unanticipated repair steps are necessary whenever a workflow is going to fail and a simple addition or substitution of tasks is not sufficient. This becomes true if those actions have side-effects that need to be repaired or avoided as well and is also necessary whenever a clean state cannot be reached without major modifications. Unanticipated repair steps may further include the continuation at another branch or position within the actual workflow (i.e. manipulating the Thread-of-Control), the substitution of parts or the complete replacement of the actual workflow as well as changes to the workflow context.

Proactive injection of activities. An extra activity (or block of activities) has to be added to the workflow due to runtime reasons. This can be necessary whenever the successful execution of the workflow is at risk and failures can be avoided through additional tasks. For an example see Section 4.

Modifying attributes of an existing workflow. Changes to the workflow context represent the least invasive modification. Two ways of using this interface are possible: the modification of an endpoint to replace external services (e.g. to comply with SLAs) or the supervision of workflow variables. That means dynamic control of loops allows for the insertion of additional iterations until data quality is sufficient (e.g. in machine learning applications). Modifications may depend on progress as well as runtime parameters not available at design-time.

There is no restriction for a maximum amount of different handlers that can be included into the execution process. To interact with the handlers, the WEE solely relies on the handler wrapper. It is the responsibility of the handler wrapper to embed / use the different handlers. In the process of execution, it is possible to replace a handler or queue multiple handlers for a single purpose.

When processing a workflow, different types of handlers may be included (see also Fig. 3):

Call Handlers are services that represent that actual functionality requested by the workflow. E.g. when the workflow wants two numbers to be summed up, it delegates this task to handler wrapper, which in turn calls the service that is intended to do so. Services are selected by passing information from the workflow to the handler wrapper. Information about call handlers (which provide functionality in the form of activities to the workflow) is part of the workflow description.

Monitoring Handlers are being informed about the process of execution. Logging functionality is one of the main applications of these handlers. They also supervise the execution to comply with Quality of Service (QoS) properties or Service Level Agreements (SLAs). If problems

arise or are anticipated, the handler may delegate the problem-solving to a repair handler and provide the necessary information [16].

Repair Handlers are invoked to bring a failing workflow back into a clean state. For this purpose, they use the intervention interface of the WEE and can apply specific repair strategies according to the problem.

Security Handler supervise the workflow execution in terms of security aspects. They grant or deny access to resources or services.

With WEE focusing solely on the control flow and the integration of different handlers through the handler wrapper, a network of modules is instructed to process the execution. Each module has a clearly specified role and can be replaced by different versions, thus fostering a loose coupling and a network-based execution.

6. Conclusion

In this paper, we presented an architecture for a modularized workflow execution engine. The WEE implementation is currently supporting all the scenarios presented in this paper, and is currently extended by new monitoring and repair handlers to show the flexibility of the architecture. Future research will include the presentation of this ongoing efforts to support dynamic business processes.

Acknowledgment

This work is partly supported by the Austrian Grid (Phase 2) project funded by the Austrian /bm:bwk (Federal Ministry for Education, Science and Culture), contract GZ BMWF-10.220/0002-II/10/2007.

References

[1] S. Modafferi, E. Mussi, and B. Pernici, "SH-BPEL: a self-healing plug-in for Ws-BPEL engines," in *Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006)*, pp. 48–53, ACM New York, NY, USA, 2006.

[2] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service," *Computer*, vol. 36, no. 10, pp. 38–44, 2003.

[3] L. L. C. ActiveBPEL, "ActiveBPEL-BPEL execution engine." http://www.activebpel.org/, 2009. [Last accessed: 10.06.2009].

[4] Oracle, "Oracle process manager." http://www.oracle.com/technology/products/ias/bpel/index.html, 2009. [Last accessed: 10.06.2009].

[5] F. Curbera, M. Duftler, R. Khalaf, N. Mukhi, W. Nagy, and S. Weerawarana, "BPWS4J." http://www.alphaworks.ibm.com/tech/bpws4j, 2002. [Last accessed: 10.06.2009].

[6] G. Stuermer, J. Mangler, and E. Schikuta, "A domain specific language and workflow execution engine to enable dynamic workflows," in *Proceedings of the 1st International Workshop on Workflow Management in Service and Cloud Computing*, IEEE Computer Society Press, 2009.

[7] B. Schneier, "Cloud computing." http://www.schneier.com/blog/archives/2009/06/cloud_computing.html, 2009. [Last accessed: 10.06.2009].

[8] C. Hagen and G. Alonso, "Exception handling in workflow management systems," *IEEE Transactions on software engineering*, vol. 26, no. 10, pp. 943–958, 2000.

[9] J. Eder and A. Tahamtan, "Temporal consistency of view based interorganizational workflows," *Proc. of UNISCON*, vol. 8, 2008.

[10] M. Bichier and K. Lin, "Service-oriented computing," *Computer*, vol. 39, no. 3, pp. 99–101, 2006.

[11] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distributed and parallel Databases*, vol. 3, no. 2, pp. 119–153, 1995.

[12] M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst, "Facilitating flexibility and dynamic exception handling in workflows through worklets," in *Proceedings of the CAiSE*, vol. 5, p. 4550, 2005.

[13] S. Rinderle, M. Reichert, and P. Dadam, "Correctness criteria for dynamic changes in workflow systemsa survey," *Data & Knowledge Engineering*, vol. 50, no. 1, pp. 9–34, 2004.

[14] S. W. Sadiq, O. Marjanovic, and M. E. Orlowska, "Managing change and time in dynamic workflow processes," *International Journal of Cooperative Information Systems*, vol. 9, no. 1-2, pp. 93–116, 2000.

[15] M. Reichert and P. Dadam, "ADEPT flexsupporting dynamic changes of workflows without losing control," *Journal of Intelligent Information Systems*, vol. 10, no. 2, pp. 93–129, 1998.

[16] J. Eder, J. Mangler, E. Mussi, and P. Pernici, "Using stateful activities to facilitate monitoring and repair in workflow choreographies," in *International Workshop on Self Healing Web Services*, (Los Angeles, CA, USA), 2009.

A grid-enabled solver for the fluid-structure interaction (FSI) problem

Ulrich Langer* and Huidong Yang† and Walter Zulehner‡

Abstract. *In this paper, we described a grid-enabled solver for the fluid-structure interaction (FSI) problem. We use a numerical method such that the whole problem is reduced to the interface equation by requring solving the fluid and the structure sub-problems which can be solved independently on each grid node. A newly designed Client/Server model under the grid environment is discussed.*

Fluid-structure interaction problems arise in many application fields such as flows around elastic structures or blood flow problems in arteries. The method used in this paper for solving such a problem is based on a reduction to an equation at the interface. This interface equation is solved by a Newton iteration. This simulation usually leads to a problem in large scale such that it has to be solved use powerful machines.

As we might see today, under the grid environment [5, 2, 4], many large scale numerical simulations can be resolved in a more efficient way because of its huge computational and memory storage resources. In our previous work, we developed a grid enabled solver in computational fluid dynamics (CFD) based on the so-called separating technique which has been tested under the Austrian Grid environment [1]*, see [8, 7, 6].*

One important issue in the numerical simulation of this nonlinear coupled FSI problem is to develop algorithms based on efficient, robust and fast solvers for each of the sub-problems (fluid and structure), which are the main costs of this type of algorithms and can be distributed and parallelized to many processors under the grid environment. We employ three distinct grid nodes. The master node is responsible for gathering and redistributing data, and synchronizing the process in each nonlinear iteration. With the Newton algorithm developed in [1], only a small amount of data (the updated displacement at the interface Γ_0*) will be transferred among the master and slave nodes during the process. The other two slave nodes will solve the fluid and structure sub-problems. In principle, each of these sub-problems can be parallelized in each node.*

Among these nodes, we need to define a flexible and secure interface for data transfer. In the next sections, we will present the mathematical background related to the solver designing ideas under the grid environment and data transferring interface on the memory level using Globus_IO secure channels [3], and the grid-enabled Client/Server model [8, 7].

*Institute of Computational Mathematics, Johannes Kepler University Linz, Linz, Austria, e-mail: ulanger@numa.uni-linz.ac.at, http://www.numa.uni-linz.ac.at/∼ulanger/.

†Institute of Computational Mathematics, Johannes Kepler University Linz, Linz, Austria, e-mail: huidong@numa.uni-linz.ac.at, http://www.numa.uni-linz.ac.at/∼zulehner/.

‡Institute of Computational Mathematics, Johannes Kepler University Linz, Linz, Austria, e-mail: huidong@numa.uni-linz.ac.at, http://www.numa.uni-linz.ac.at/∼huidong/.

[1] http://www.austriangrid.at/

However, this is only the first draft implementation which does not want to achieve optimal performance and thus serves as a proof of concept study. Starting from this general framework, we can go further. For instance, the sub-problems can do their jobs in parallel on different nodes such that they can utilize the computing and memory resources in grid nodes. Future work could concentrate on improving the performance and investigating the scalability of the method.

References

[1] S. Deparis, M. Discacciati, G. Fourestey, and A. Quarteroni. Fluid-structure algorithms based on Steklov-poincaré operators. *Comput. Methods Appl. Mech. Engrg*, 195:5797–5812, 2006.

[2] C. Fellenstein and J. Joseph. *Grid Computing*. Prentice Hall Ptr, 2003.

[3] L. Ferreira, V. Berstis, and J. Armstrong. *Introduction to Grid Computing with Globus*. IBM Corp, 2003.

[4] I. Foster and C. Kesselman. *Grid 2 Blueprint for a New Computing*, volume 13 of *Elsevier Series in Grid Computing*. Morgan Kaufmann Pub, 2003.

[5] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *High Performance Computing Applications*, 15(3):200–222, 2001.

[6] U. Langer and H. Yang. A parallel solver for the 3D incompressible Navier-Stokes equations on the Austrian Grid. SFB-Report 06-12, SFB "Numerical and Symbolic Scientific Computing" F013, Johannes Kepler University of Linz, 2006.

[7] U. Langer, W. Zulehner, H. Yang, and M. Baumgartner. GStokes: A Grid-enabled Solver for the 3D Stokes/Navier-Stokes System on Hybrid Meshes. In *ISPDC*, pages 377–382, 2007.

[8] H. Yang, W. Zulehner, U. Langer, and M. Baumgartner. A robust PDE solver for the 3D Stokes/Navier-Stokes systems on the grid environment. In *GRID*, pages 145–152, 2007.

Parallel Algebraic Multigrid on General Purpose GPUs

Gundolf Haase* and Manfred Liebmann† and Gernot Plank‡ and Craig Douglas†

Abstract. *We will present results for a parallel algebraic multigrid that runs on GPUs. The system matrix of the system of equations under consideration originates from a 3D unstructured mesh from medical application. The overall speedup of the GPU implementation reflects the much higher memory bandwidth on the GPU in comparison to the CPU.*

1. Introduction

With the introduction of IEEE 754 double precision floating point support on Nvidia's GT200 series graphics processing units (GPUs) the implementation of key elements of numerical algorithms used in scientific computing became feasible. The paper will discuss the design and implementation of a parallel preconditioned conjugate gradient algorithm with an algebraic multigrid preconditioner[4][8][6][5]. The so called PCG-AMG algorithm is typically used to solve elliptic partial differential equation. This type of linear algebra problem appears in many types of applications from fluid dynamics, solving the Navier-Stokes equation, electromagnetic problems, solving variations of Maxwell's equation, and other potential type problems. We present a comparison of the PCG-AMG algorithm on different hardware platforms i.e typical cluster computer with Infiniband and Ethernet interconnect and multi–GPU compute servers. A benchmark of an elliptic model problem using an unstructured 3D grid derived from a virtual heart simulation [7] shows that a single GPU server delivers up to 100x the performance of a typical server CPU core and thus easily outperforms whole cluster computers. Thus the focus of the paper is on aspects of GPUs computing that are important for the implementation of the PCG-AMG algorithm on graphics hardware.

2. Parallelization on Graphics Processing Units

2.1. An Introduction to GPU Computing

Although the idea of general purpose computing in graphics processing units has been proposed several years ago and has been supported by ATI with the high level BROOKS language and NVIDIA with various approaches to shader languages, the technology never really took off and was more a curiosity than useful tool. The main technical reason for this failure was the fundamental limitation of the programming model tied to the *stream computing model*. In layman terms this means that

*Institute for Mathematics and Scientific Computing, University of Graz, email: gundolf.haase@uni-graz.at
†Department of Mathematics, University of Wyoming, email: manfred.liebmann@uni-graz.at
‡Computing Laboratory, Oxford University, email: gernot.plank@comlab.ox.ac.uk

algorithms are limited to *sequential* access of data. Although the stream computing model is natural for 3D graphics, where thousands of triangles are processed piece by piece by the graphics pipeline, but it is a severe limitation if one wants to implement general purpose algorithms.

About one year ago NVIDIA changed the picture of general purpose computing in a radical way by allowing random memory access to data structures for all computational resources on the GPU. Enabling this technology required hardware changes to the processing cores on the GPU and also the design of the *Compute Unified Device Architecture* CUDA software technology [1]. With the hardware and software in place, the processing cores on the GPU, also called thread processors, behave essentially like traditional CPUs. From a higher level point of view a GPU can now be thought of as a shared memory machine with hundreds of processing cores.

The CUDA toolkit provides C/C++ compilers and runtime libraries that make it possible to compile standard C/C++ code to run on the GPU, where the GPU acts as a coprocessor to the host computer that holds the GPU boards. The CUDA programming model is build on the concept of running a program on a server where compute intensive functions are handled by the GPU boards. For example a high end graphics board today contains a GPU with several hundred thread processors and up to 1 GB of high speed memory. NVIDIA even provides the TESLA line of pure compute boards that use the same GPUs but features up to 4 GB onboard memory.

The CUDA technology allows an *incremental* software design approach. This means that a given C/C++ or Fortran code can be extended step by step using computational kernels that run on the GPU using it as a coprocessor. Often the performance of an algorithm is determined by only a few computational kernels, where most of the compute time is spent. Replacing these few kernels with GPU optimized implementations immediately gives the potential of speeding up the whole algorithm with minimal effort.

Typically well implemented GPU kernels are 10x - 100x times faster than their CPU counterparts. This can be attributed to the hundreds of processing cores on the GPU and the very fast onboard memory, that is typically an order of magnitude faster than typical server memory. Although one limitation of the GPU coprocessing model has to be clearly stated here and that is to get the data on the GPU board for fast processing. GPU kernels work only with data in the onboard memory, so data has to be transferred in and out of the GPU. The GPU boards are connected to the host computer via the PCIE bus. The speed of the PCIE bus is comparable to the speed of typical memory chips in the host computer and gives about 5 GB/s of memory copy performance. Compared to the GPU memory with a memory copy performance of 120 GB/s this is still slow.

From the general analysis above it is clear that it is very important to limit the data transfer between the GPU and the host computer. To use the full potential of the GPU all data structures required for the computational kernel must be kept in GPU memory.

2.2. Sparse Matrix-Vector Multiplication Kernel

Maybe the most important computational kernel for the PCG-AMG solver is the sparse matrix-vector multiplication. This kernel is the very key to the performance of the solver and a great amount of time has been spent designing and implementing the kernel. Due to the coalescing restriction, a specific memory access pattern restriction, it is not possible to use the standard compressed row storage data format on the GPU. Any attempt will lead to very bad performance, since there is no

natural blocking within the data structure to implement coalescing. With this insight gained from extensive performance tests with various variations of the CRS algorithms a new data structure is needed. It is important to keep the complexity of data structure conversion low. With this in mind an interleaved CRS data structure turned out to be the best choice. A comparison of different sparse matrix-vector multiplication kernels can be found in [3] and [2].

Aside from the choice of the data structure the standard C/C++ CRS algorithm transfers naturally with minor modification to the GPU. The code listing in the next section shows the algorithm in detail.

2.3. Interleaved Compressed Row Storage Data Format

In accordance with the coalescing memory access needed to get the full GPU performance we had to rearrange the CRS format such that L consecutive matrix rows are interleaved. As a consequence, the matrix elements of a row are accessed with stride$L$ and all matrix rows in this block of L rows with less elements than the others have to filled with (unused) dummy data.

The compressed row storage data format consists of four blocks of data: `cnt`, `dsp`, `col`, and `ele`. The vectors `cnt` and `dsp` store the number of nonzero elements per row and the offset to the first element of a row. These data structures are necessary to execute the sparse scalar products, that build up the matrix multiplication, in parallel. Specifically the rows of the column indices `col` and matrix entries `ele` are stored interleaved in blocks of $L = 64$ rows. Further special texture instructions as `cudaBindTexture` and `tex1Dfetch` are used in the GPU kernel to speed up the random access to the right hand side via the texture cache.

```
struct linear_operator_params {
    const int *cnt;                 //  matrix in
    const int *dsp;                 //     interleaved (stride L)
    const int *col;                 //     CRS
    const double *ele;              //     format
    const double *u;                //  right hand side vector to multiply with
    double *v;                      //  resulting vector
    int n;
};
                                    //  kernel function running on the GPU
texture<int2> tex_u1;
__global__ void __device_linear_operator(struct linear_operator_params parms)
{
    unsigned int j = N * blockIdx.x + threadIdx.x;    // identify matrix row for  this thread
    if(j < parms.n)                                   // valid row number?
    {                                                 //   (more threads than rows are possible)
        unsigned int blkStart = parms.dsp[j];
        unsigned int blkStop = blkStart + L * parms.cnt[j];
                                        //    matrix elements of row are stored with stride L
        double s = 0.0;
        for(unsigned int i = blkStart; i < blkStop; i += L)
        {
            unsigned int q = parms.col[i];            //  column index
            double a = parms.ele[i];                  //  matrix element
            int2 c = tex1Dfetch(tex_u1, q);           //  get vector entry from texture cache
            double b = __hiloint2double(c.y, c.x);    //     convert this entry to double
            s += a * b;
        }
        parms.v[j] = s;                               //  store the result
    }
}
```

```
extern "C" {                        //  host function running on the CPU
void _device_linear_operator(int *cnt, int *dsp, int *col, double *ele,
    int l, int m, int n, double *u, double *v)
{                                                   // bind right hand side to texture
    cudaBindTexture(0, tex_u1, (int2*)u, sizeof(double) * m);

    struct linear_operator_params parms;            //  setup data for operation
    parms.cnt = cnt;
    parms.dsp = dsp;
    parms.col = col;
    parms.ele = ele;
    parms.u = u;
    parms.v = v;
    parms.n = n;
    __device_linear_operator<<< (n + N - 1)/N, N >>>(parms);  // perform matrix-vector multiplication

    cudaUnbindTexture(tex_u1);
}
}
```

3. Benchmarks for Elliptic Model Problems

3.1. Performance Results

We use an unstructured 3D mesh from the virtual heart problem [7] as basis for the numerical experiments. The resulting sparse finite element matrix consist of $862,515$ rows with $12,795,209$ non-zero entries. We compare the performance of the complete PCG-AMG algorithm running on a GPU server with different cluster computers and servers. The benchmark times 64 iterations of the PCG-AMG algorithm, see Tab. 1 for details. All computations use double precision arithmetics.

#cores	kepler	liebmann	mgser1	gtx	gpusrv1	mgser1	gtx	gpusrv1
	code on CPU–cores					code on GPU–cores		
1	29.239	30.253	22.615	17.026	9.607	**1.217**	1.016	1.238
2	14.428	15.954	11.999	9.709	5.662		0.612	0.726
4	7.305	7.544	8.490	**6.562**	**3.885**		**0.367**	0.409
8	3.607	4.054	**8.226**					**0.284**
16	1.909	**3.493**						
32	**1.167**							

Table 1. Solver times in seconds for the servers: kepler (AMD Opteron Infiniband Cluster), liebmann (AMD Opteron Quad Socket Shared Memory Server), mgser1 (Intel Xeon Dual Socket Server), gtx (AMD Phenom 9950 Server), gpusrv1 (Intel Core i7 Server) and for the GPU servers: mgser1 (1x Tesla C1060), gtx (4x Geforce GTX 280), gpusrv1 (4x Geforce GTX 295)

We compare the performance of the Infiniband cluster computer *kepler* with 16 dual–socket single core Opteron 2.2GHz nodes and the quad–socket shared memory server *liebmann* with quad–core Opteron 1.9GHz processors and the dual-socket quad-core Xeon 2.6GHz server *mgser1* and the single socket quad-core Phenom 2.6GHz server *gtx* and the single socket quad-core Intel Core i7 3.2GHz server *gpusrv1* with three GPU servers setups. The first GPU server *mgser1* has a single Tesla C1060 board with 4GB of memory and 240 processing cores. The second custom built GPU server *gtx* has four Nvidia GTX 280 boards with 1GB memory each and 960 processing cores. The third most powerful GPU server *gpusrv1* has four Nvidia GTX295 dual GPU boards with 1.8GB memory each and 1920 processing cores.

The performance of the GPU server *gpusrv1* with all eight GPUs active is about 100x of a single Opteron core on the cluster computer or shared memory machine. Even compared to the fastest Intel Core i7 CPU with all four cores active the GPU server is more than 13x faster. A single Tesla board is essentially as fast as the whole Infiniband cluster with all 32 CPUs active and the fastest GPU server beats the Infiniband cluster by a factor of four.

4. Conclusions

The benchmark with an AMG preconditioned conjugate gradient solver shows the huge potential of GPU computing and of many–core chip architectures in general. We would like to emphasize that our algorithms are performed on unstructured matrices from a 3D discretization of a real world example.

The 10x – 100x price/performance advantage of GPU servers in comparison with traditional parallel computers is very attractive, especially when the algorithms are limited by the memory bandwidth as in our case. The CUDA programming interface provides an easy to use interface for GPU programming although the knowledge of internal details is necessary to achieve the full performance.

References

[1] Nvidia CUDA. http://www.nvidia.com/cuda/.

[2] M. M. Baskaran and R Bordawekar. Optimizing sparse matrix-vector multiplication on GPUs. *IBM Technical Report RC24704*, 2008.

[3] N. Bell and M. Garland. Efficient sparse matrix-vector multiplication on CUDA. *NVIDIA Technical Report NVR-2008-004*, 2008.

[4] C.C. Douglas, G. Haase, and U. Langer. *A Tutorial on Elliptic Pde Solvers and Their Parallelization*. Society for Industrial and Applied Mathematics, 2003.

[5] Dominik Göddeke, Robert Strzodka, Jamal Mohd-Yusof, Patrick McCormick, Hilmar Wobker, Christian Becker, and Stefan Turek. Using GPUs to improve multigrid solver performance on a cluster. *International Journal of Computational Science and Engineering*, 4(1):36–55, 2008.

[6] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. The MIT Press, Cambridge, 1999.

[7] G. Plank, M. Liebmann, R. Weber dos Santos, E.J. Vigmond, and G. Haase. Algebraic multigrid preconditioner for the cardiac bidomain model. *IEEE Transactions on Biomedical Engineering*, 54(4):585–596, 2007.

[8] J. W. Ruge and K. Stüben. Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). *Multigrid methods for integral and differential equations. The Institute of Mathematics and Its Applications Conference Series. Oxford: Clarendon Press*, pages 169–212, 1985.

Secure Scalable Video Compression for GVid

Heinz Hofbauer and Thomas Stütz and Andreas Uhl *

1. Introduction

The GVid framework and implementation has been introduced and discussed in previous work [2]. It separates the task of data generation and visualization from the comparably computationally inexpensive task of transmission and display. This separation is especially reasonable if the visual data is displayed on a computationally weak device. Mobile devices have become the most frequent computing platform for a majority of users but suffer from slower CPUs, less memory, lower resolution displays, and network connections with lower bandwidth and higher probability of connection loss. Especially these restricted computational capabilities are a convincing argument for the separation of data generation and visualization from the comparably computationally inexpensive task of transmission and display. Additionally the varying network parameters paired with a higher probability of connection loss pushes the development of scalable and error resilient formats and transmission systems for visual data. Scalable visual data formats enable simple and fast rate adaptation. At present the scalable extension of the video coding standard H.264 (SVC) has been finalized and thus an applicable scalable video compression system is now available. A different approach to implementing a scalable video format compared to the traditional design of H.264/SVC is followed by the wavelet-based MC-EZBC codec. Both schemes offer state-of-the-art scalable video compression and are therefore evaluated for the suitability as compression codecs within the GVid framework. Their compression performance is evaluated and format-specific encryption approaches are discussed together with a motivation and introduction to format-specific encryption. The major advantages of format-specific encryption schemes are the preservation of scalability in the encrypted domain, i.e. rate adaptation can still be conducted, and a potentially improved error robustness and resilience.

2. GVid: Secure Interactive Video Transmission

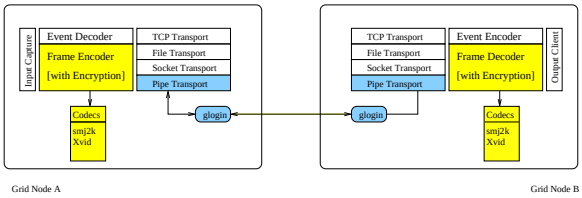


Figure 1. GVid Component Overview

The aim of GVid software design was to support as many applications as easily as possible. Therefore, several input adapters exist that are responsible for acquiring the visual data of the application.

*Department of Computer Sciences, University Salzburg, Salzburg, Austria, email: {hhofbaue, tstuetz, uhl}@cosy.sbg.ac.at

Figure 1 illustrates an overview of the GVid design. An application provides the visual data through one of the several input adapters and GVid takes care of the encoding, the secure and efficient transmission and the display of the visual data.

Currently a compression plug-in for MPEG-4 (Xvid) and JPEG2000 are integrated but new compression and security schemes can be easily integrated. Xvid does not provide a scalability thus rate adaptation or delivery of streams at different rates can not be done efficiently. A scalable video compression system would perfectly meet the requirements of efficient compression and scalability of the video format stream. The idea of the application of scalable format streams for network adaptation has been extended to in-network adaptation systems, in which adaptation is dynamically performed in the network by a MANE (media aware network element). These in-network adaptation systems offer rapid adaptation to changing network conditions as the delay for the propagation of changed network parameters is minimized.

However, the application of well-established security tools is not possible as the necessary information to perform rate-adaptation within the network is concealed and thus not available at the MANE. Thus if security and in-network adaptation are to be combined, format-specific encryption schemes, that preserve the information necessary for rate adaptation, are needed.

3. State-of-the-Art Scalable Video Compression

In the following two scalable video compression systems are presented, which also represent two different approaches to implement scalable video coding.

SVC follows the traditional design of layered video coding while MC-EZBC is a t+2D wavelet-based video codec with motion-compensated temporal filtering.

3.1. H.264/SVC

A major design requirement for SVC has been the backwards compatibility to the existing H.264/AVC [1]. Thus SVC format streams are valid H.264/AVC format streams (format-compliant with respect to the non-scalable H.264/AVC format) and thus decodeable by H.264/AVC compliant decoders. An SVC format stream contains a base layer and one or more enhancement layers each may augment the user experience in one of three dimensions (temporal/spatial/quality). A H.264/SVC stream is scalable if it contains sub streams with lower frame rates, different resolutions or qualities.

3.2. MC-EZBC

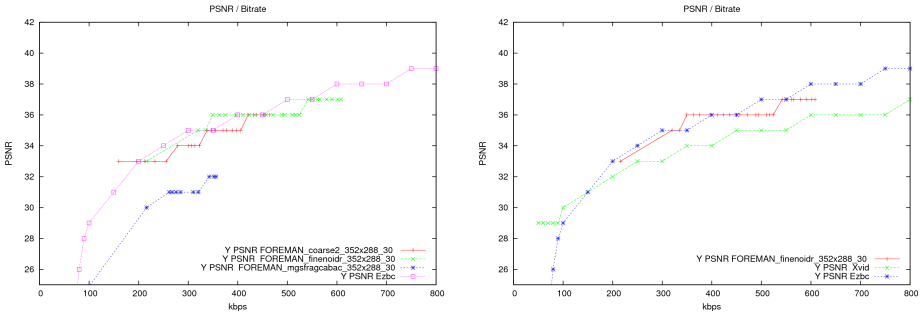
The MC-EZBC [4] coder is a t+2D wavelet coder, i.e., a wavelet transform is applied for temporal decomposition as well as for spatial decomposition. The abbreviation t+2D implies that the temporal decomposition combined with motion estimation is applied before the spatial decomposition (both apply pyramidal decomposition structures). The 9/7 CDF (Cohen-Daubechies-Feauveau) wavelet filters are applied for spatial decomposition, while temporal decomposition is conducted with the CDF 5/3 wavelet filters. Furthermore adaptive prediction techniques are employed, i.e. motion compensation and neighbourhood prediction.

A MC-EZBC stream is by design temporally, spatially and SNR (signal to noise ratio) scalable.

3.3. Performance Evaluation

For more extensive and exhaustive treatments on the compression performance of these codecs the reader is referred to [3]. MC-EZBC's compression performance is at least equal to H.264/SVC, see fig. 2(a). It has to be noted, that the results for H.264/SVC have been obtained with the reference software JSVM and that other implementations may offer better compression performance. If both codecs are compared to state-of-the-art MPEG-4(Xvid), the clear resume is that both perform significantly better for a broad range of bitrates, see fi. 2(b).

The advantage of the MC-EZBC is its higher flexibility in terms of possible extraction points; beneficial if fine grained rate adaptation is to be performed. There are several arguments for H.264/SVC: It is backwards-compatible to H.264, which allows the base layer to be decoded with a compliant H.264 decoder. The base layer can be encoded such that decoding has a very low computational complexity. In conclusion, MC-EZBC offers better rate adaptation, but H.264/SVC provides important features like backwards compatibility.



(a) MC-EZBC compared to H.264/SVC (b) Comparison of Xvid, MC-EZBC and H.264/SVC

4. Format-Specific Encryption Schemes

Format-specific scalability-preserving encryption schemes are necessary in order to combine efficient transmission and confidentiality. In the following format-specific encryption schemes are discussed for H.264/SVC and MC-EZBC. Both schemes preserve format-compliance,i.e., the decoder does not crash and offer efficient encryption of the coded video data, while preserving the scalability. Thus both schemes are well-suited for the integration in the GVid framework.

4.1. H.264/SVC-Specific Encryption

In order to preserve the scalability of the H.264/SVC stream, the information to which dependency layer, temporal layer and quality layer a NAL (network abstaction layer) unit contributes, i.e., a part of the SVC extended NAL unit header, has to be preserved. Thus only encrypting the NAL unit body preserves scalability. However, straight-forward conventional encryption of the NAL unit body is problematic, as NAL unit bodies obey certain syntax rules. Namely marker sequences, that e.g., signal the beginning and the end of a NAL unit, are forbidden. Thus conventional encryption of NAL unit bodies is likely to break the system at some point, e.g., if the H.264/SVC byte-stream format is used and a marker sequence is accidentally generated in a NAL unit body, the entire synchronization is lost.

4.2. MC-EZBC-Specific Encryption

As format-specific encryption for MC-EZBC heavily relies on its bitstream format, we start the with a discussion of the MC-EZBC format. The bitstream starts with a main header followed by GOP (group of picture) sizes followed by coded data of sequential GOPs. Each GOP is lead by a header, giving scene change information, followed by the motion field and coded image data. Both motion field and image data are ordered by frame (temporal resolution). The image data of a frame is also arranged from lowest to highest resolution and a spatial decomposition of a frame is grouped together as a basic image data unit (we will call them chunks from now on). Each chunk is preceded by a leading header defining the length of the chunk and groups all chroma information of a given decomposition level. The image data in a chunk is ordered by importance regarding SNR scalability and is the result of a bitplane coder. This enables SNR scaling through truncation of image data.

Only when all headers, including chunk headers, and GOP size information are kept intact the whole bitstream can subsequently be parsed correctly, which is of importance for the preservation of scalability. Additionally the motion vector data is coded differently from the image data; the length of the motion vector data is not explicitly signalled, but it has to be determined by arithmetic decoding. Thus headers and motion vector data will not be encrypted in our encryption scheme, but solely the coded image data.

5. Conclusion and Future Work

We have evaluated two state-of-the-art scalable video compression systems, namely H.264/SVC and MC-EZBC, for their suitability as compression codecs in the GVid framework. Their compression performance is competitive to conventional video compression systems, such as the MPEG-4 implementation Xvid. Their scalable format streams offer improved performance for multiple application scenarios. As the application of conventional security tools for confidentiality circumvent the advantages of scalable compression systems, format-specific encryption tools are necessary. For both H.264/SVC and MC-EZBC format-specific and even format-compliant encryption schemes have been proposed and discussed. Both encryption schemes meet the requirements well and can be recommended for integration in the GVid framework.

References

[1] ITU-T H.264. Advanced video coding for generic audivisual services, November 2007.

[2] T. Köckerbauer, M. Polak, T. Stütz, and A. Uhl. GVid - video coding and encryption for advanced Grid visualization. In J. Volkert, T. Fahringer, D. Kranzlmüller, and W. Schreiner, editors, *Proceedings of the 1st Austrian Grid Symposium*, volume 210 of *books@ocg.at*, pages 204–218, Schloss Hagenberg, Austria, 2006. Austrian Computer Society.

[3] M. Wien, H. Schwarz, and T. Oelbaum. Performance analysis of SVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1194–1203, September 2007.

[4] Y. Wu, A. Golwelkar, and J. W. Woods. MC-EZBC video proposal from Rensselaer Polytechnic Institute. *ISO/IEC JTC1/SC29/WG11, MPEG2004/M10569/S15*, March 2004.

Grid-based Scientific Dataspace Support Platform for Breath Gas Analysis

Ibrahim Elsayed,[*] Thomas Ludescher,[†] Philip Masser[†], Thomas Feilhauer,[†] and Peter Brezany[*]

Abstract. *A Scientific Dataspace Support Platform aims at providing associated mechanisms for managing semantically rich relationships among scientific data sources (primary data) and its corresponding findings (derived data), which result from a set of activities defining concrete preprocessing and analysis methods (background data) that were applied on a source dataset. Breath gas analysis is an emerging new scientific field with a large scientific community spread all over the world and with a promising significant impact on many application domains. The breath gas analysis community is investigating and screening for hundreds of compounds in the exhaled breath, resulting in an daily increase of scientific data that is stored locally at the breath gas research institutions waiting for further experimentation also from scientists belonging to other research centers. This paper describes an implementation of the scientific dataspace paradigm for breath gas analysis on the Grid offering the needed services in order to organize and retrieve scientific data in a dataspace. The main contribution is the development of a semantic relationship model for distributed participants of a dataspace and its evaluation within a real world e-Science application.*

1. Introduction

The idea of a future data management paradigm called *dataspace* was introduced by Franklin et al. [3] and also addressed by authors of this paper in [4, 5]. The goal is to manage a dataspace, rather than a database. Dataspaces are modeled as participants (datasets) and relationships. The concepts of a scientific dataspace paradigm are described in [5]. It introduces the *e-Science lifecycle*, which represents *a domain independent ontology-based iterative model, tracing semantics about procedures in e-Science applications. Iterations of the model - so called e-Science life cycles - organized as instances of the e-Science life cycle ontology, are feeding a dataspace, allowing the dataspace to evolve and grow into a valuable, intelligent, and semantically rich space of scientific data.* The major role of the e-Science lifecycle ontology[1] is to describe and semantically enrich the existing relationship among primary and derived data sets in e-Science applications. This work contributes to the development of support platforms for scientific dataspaces. Main focus is to develop, and evaluate a relationship model for creation, representation, and maintaining of semantically enriched relationships among distributed dataspace participants.

[*]Department of Scientific Computing, University of Vienna, Nordbergstrasse 15/C/3, A-1090 Vienna, Austria email: {brezany|elsayed}@par.univie.ac.at

[†]Research Center for Process and Product Engineering, University of Applied Sciences, Hochschulstrasse 1, A-6850 Dornbirn, Austria email: {thomas.ludescher|thomas.feilhauer|philip.masser}@fhv.at

[1]www.gridminer.org/e-sciencelifecycle

The analytical instruments and techniques used by the growing international community of gas researchers include diverse mass spectrometers as well as various statistical and data mining techniques supporting identification of specific markers. Currently, during the investigations – mainly performed using the common MATLAB language and computing environment – a large number of new different sampling and analytical techniques for breath gas measurements are being developed.

Breath gas analysis scientific dataspaces are providing collaborating scientists and institutions (a) access to distributed breath gas data and analytical resources collected and developed at different research institutions around the world and (b) to easily contribute to and leverage the resources of an international- and national-scale, multi-institutional environment. This will strongly support global collaborations of scientists, improve decisions and increase the chance and scope of discoveries in the breath gas research domain. *Source data* obtained from the above-mentioned analytical methods is referred to as breath gas measurement data and is saved, together with corresponding patient data, locally at each research center. This data is undertaken for significant simulation and modeling in the context of main focus of the acting research group, e.g. observation of the correlation between isoprene breath content and cholesterol level in blood [9]. The output of these analyses aims at defining a large number of predictions and might provoke further experimentation, which in turn may take days or weeks, depending on computational and human resources available. However, the resulting data called the *derived data*, that have arisen from the research task represents valuable information not only to the acting research group, but also to other groups with respect to other main focuses.

2. Breath Gas Analysis Dataspaces

Breath gas analysis dataspaces consists of a set of databases set up by a *dataspace designer* and administered by a *dataspace manager*. In particular, there are (1) *primary databases* for storing the input datasets, (2) *background databases* for storing the analytical methods used to analyze the input dataset i.e. MATLAB M-files, (3) *derived databases* for storing the results of analyses tasks, and (4) *other databases* i.e. *volatomics database*, which contains data from studies of exhaled breath as well as other sources of endogenously-derived gases such as skin, urine, faeces, and flatulence. In addition there are special databases set up for storing the instances of the e-Science lifecycle ontology, which are defined in RDF [12]. Because security and privacy issues are of utmost importance within applications in the breath gas analysis research domain as patient data is involved and due to legal requirements on such highly sensitive personal data, all participating databases including the RDF store are isolated, monitored, and restricted to a single point of entry using the OGSA-DAI interface, hence implement strict access control. OGSA-DAI is the de facto standard for data access and integration on the Grid for relational and xml data as well as file resources. Details about the access control mechanisms and the security considerations in general are described in [1, 2].

2.1. Architecture - Prototype Implementation

A first prototype is being implemented in order to proof the design of the architecture and the use cases described in [6] and [7], respectively. A three-tier architecture is chosen in order to lighten the front-end by moving the application logic to the middle tier, which communicates with the third tier, called dataspace-tier. Figure 1 illustrates the three tiers, its components and interactions.

Dataspace services implementing special OGSA-DAI activities communicate with the scientific dataspace and its corresponding sub-dataspaces. They implement the functionality needed in order to (1)

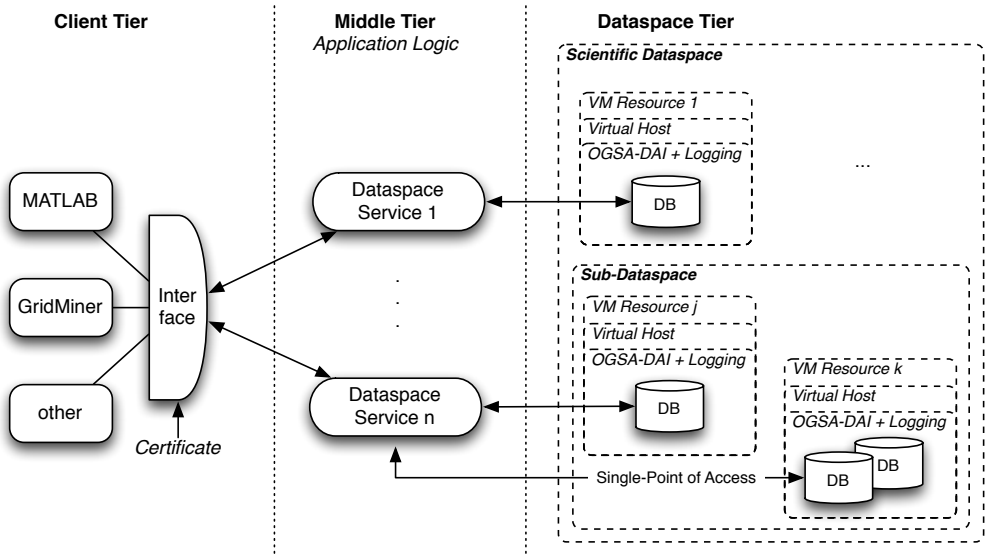


Figure 1. Three-tier architecture of the Scientific Dataspace Support Plattform

publish data belonging to a scientific experiment into the dataspace, (2) subscribe those scientific experiments and corresponding dataspace participants, (3) create, maintain and semantically enrich realtionships among those participants, and (4) search and query the scientific resources based on semantic web technologies. On the client tier different clients are interacting via a common interface with the dataspace services. Breath gas researcher are mainly using Matlab for their experimentations, however GridMiner [13] or any other data analysis tool could be used instead.

3. Related Work

The concept of dataspaces gave rise to new data management challenges and influenced many data management and personal information management approaches. This extended abstract however discusses only a view dataspace related works due to limit of space. A detailed study was elaborated in [6].

iMeMex - Personal Dataspace iMeMex [8] provides data management functionalities, such as querying, updating, performing backup and recovery operations. All data is presented using a single graph data model and queried using an own query language, called the iMeMex Query Language (iQL). Special converters convert the contents of data sources into the internal graph structure. Core idea of iMeMex is a logical layer called Resource View Layer that abstracts from underlying subsystems and data sources. A Data Source Proxy connects to the data sources. The Data Source Proxy provides plugins for file systems, IMAP email servers and RSS feeds, which shows that iMeMex is designed for personal information management, however not limited to it.

Other dataspace related work In addition to the above described related works, there are a number of research institutions working on dataspaces. For example Yukun et al. describe in [11] a personal dataspace management system, named OrientSpace, which implements data integration and data query functions. They introduced the CoreSpace framework, which represent a subspace of the personal dataspace containing only objects that are frequently accessed by the owner. The datamodel

used is based on the vertical data model, which takes a vector to describe attributes of an object (ObjectID, AttributeName, AttributeValue). Another dataspace management approach is proposed by Lei et al. in [10]. It introduces the galaxy data model, which is an extension of the iMeMex Data Model in order to better consider security issues, primarily access policies.

4. Conclusions

In this extended abstract we have introduced a novel scientific data management paradigm based on the concepts of dataspaces and Grid technology, named *Scientific Dataspace Support Platform*. Main contribution is a relationship model for creation, representation, and advanced searching of semantically rich relationships among dataspace participants. In cooperation with Austrian Grid partners we have started to develop the system in the context of a real world e-Science application, that is breath gas analysis. This application-driven development process will help us in order to evaluate a first prototype and following to improve and deploy it for real usage.

References

[1] M. Descher et al. Position paper: Secure infrastructure for scientific data life cycle management. In Proceedings of International Conference on Availability, Reliability and Security, IEEE, 2009.

[2] M. Descher et al. Retaining data control to the client in infrastructure clouds. In Proceedings of International Conference on Availability, Reliability and Security, IEEE, 2009.

[3] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: A new abstraction for information management. ACM SIGMOD, December 2005.

[4] I. Elsayed, P. Brezany, and A M. Tjoa. Towards realization of dataspaces. Proceedings of International Conference on Database and Expert Systems Applications (DEXA), 2006.

[5] I. Elsayed, A. Muslimovic, and P. Brezany. Intelligent dataspaces for e-science. CIMMACS '08, 2008

[6] I. Elsayed, A. Muslimovic, and P. Brezany. Dataspace Paradigms: State of the Art in Dataspace Paradigms and First Proposal of Dataspace Model. Technical Report AG-D7a-1-2008_v1.4, 2008

[7] I. Elsayed, et al. Towards Realization of Scientific Dataspaces for the Breath Gas Analysis Scientific Community. Internal yet unpublished document, 2009

[8] J.-P. Dittrich. iMeMex: A platform for personal dataspace management. Proceedings of SIGIR Workshop on Personal Information Management (PIM), 2006.

[9] I. Kushch et al. Breath isoprene - aspects of normal physiology related to age, gender and cholesterol profile as determined in a proton transfer reaction mass spectrometry study. Clin Chem Lab Med. 2008;46(7):1011-8.

[10] L. Jin, Y. Zhang, and X. Ye. An extensible data model with security support for dataspace management. In *HPCC*, pages 556–563, 2008.

[11] Y. Li and X. Meng. Research on personal dataspace management. In *IDAR*, pages 7–12, 2008.

[12] W3C. Resource description framework (RDF). http://www.w3.org/RDF/, 2003.

[13] P. Brezany and I. Janciak, and A M. Tjoa. GridMiner: A Fundamental Infrastructure for Building Intelligent Grid Systems. WI '05: Proceedings of the International Conference on Web Intelligence, 2005.