Exploring OGSA Interoperability with LCG-based Production Grids for Biomedical Applications

Alfredo Tirado-Ramos, Derek Groen, Peter M.A. Sloot

Faculty of Science, Section Computational Science,
University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
emails: [alfredo, djgroen, sloot]@science.uva.nl

Abstract

Interconnected Grids that communicate and transfer data and computation among different infrastructures allow scientists to create distributed systems that support complex computational science. Large production Grids that support biomedical applications, for instance, start now to become more common, connecting infrastructures and services via standard architectures such as the new Open Grid Services Architecture (OGSA) or the more mature Large Hadron Collider Computing Grid (LCG). Complex biomedical applications, nevertheless, require some degree of structural interoperability on top of available Grid connectivity in order to access data distributed among different Grid infrastructures and to submit computational jobs transparently. We present our results after performing a series of data transfer and computational job submission using a blood-flow simulation application within our biomedical problem solving environment, using Globus Toolkit 4 (OGSA) services interoperating with our current extended LCG 2.6 production-level infrastructure. We study this hybrid approach to Grid-based biomedical data access and computation, and present our conclusions.

Keywords: Computational Grids, LCG, OGSA, GT4, Interoperability, Biomedical Applications

1 Introduction

High-performance Grid distributed computing continues to evolve and to become a standard tool for data access and computational job submission in scientific organizations. Virtual scientific communities and organizations are currently being created and maintained which support large, distributed and diverse information sources [1]. Access to such resources is inherently complex, and is aggravated by structural heterogeinity of both the resources and the software infrastructures that support them. In the case of biomedical applications data are usually available in heterogeneous formats and from various legacy sources, and computational job submission is often supported by infrastructure-specific frameworks [2], [3] that may be required to intercommunicate; as Rambadt et

al [4] state: "Different (Grid) projects focus on different aspects and it is only natural to combine them". Wide-area access to biomedical information and computation usually requires higher degrees of interoperability. Currently available Grid technology supports data access and computational job sumission within specific toolkits, though the diverse biomedical informatics tools that generate and consume data rarely come from within a single source or project [5], requiring resource and infrastructure interoperability in order to access resources seamlessly across Grid virtual organizations. A. Ouksel et al [6] differentiate four types of interoperability: semantic, syntactic, system, and structural. In Grid computing, while there are a number of efforts currently at work in the fields of semantic [7], [8], syntactic [9], [10], [11], [2], [3] and system [12], [13], [14] interoperability, there is still much work to do into the issues related to structural interoperability (e.g., seamless access to a set of shared infrastructure services) [15], [16]. On one hand, the Open Grid Services Architecture (OGSA) [17] is one of the main results of a global effort to advance Grid interoperability, extending the Web Services [18] paradigm to include Grid concepts, and to manage the creation and termination of resources as real services. This Service Oriented Architecture (SOA) approach to Grid computing is now implemented in the Globus Toolkit 4 (GT4) [19], one of the most popular implementations of the OGSA architecture and Web Services Resource Framework (WSRF) specification [20]. The GT4 is currently being deployed among new Grid development projects, and is expected to become a de-facto standard for Grid services. On the other hand, more mature frameworks such as the CERN's Large Hadron Collider Computing Grid (LCG) [21] have been successfully deployed by production-type Grid infrastuctures like the European CrossGrid project [3], providing scientists with a production testbed that is maintained continuously. Our Grid infrastructure is based on and supported by the CrossGrid project's testbed, including resources spread across Europe, which range from relatively small computing facilities in universities to large research computing centers. The CrossGrid testbed largely inherits from the European DataGrid (EDG) [22] experience on setup and it is fully based on LCG middleware distributions for services.

We present our results after performing a series of interoperability tests between recently added GT4 and LCG resources within our testbed for biomedical applications. We study biomedical data transfer and job submission among these infrastructures, focusing on biomedical dataset transfer times, CPU usage overhead, as well as job submission using both intra and inter cluster computational runs, and present our conclusions. The structure of the paper is as follows: Section 2 briefly describes the Grid Virtual Organization (VO) concept and the SOA-based OGSA. Section 3 elaborates on one of the most common approaches to production-type Grids, a European infrastructure based on it, and the biomedical application used for our experiments. In Section 4 we present the results of our interoperability experiments from the data and computational viewpoints, whereas Sections 5 and 6 offer a discussion on the results and future work.

2 A Service Oriented Approach to the Grid

2.1 The Open Grid Services Architecture

The Open Grid Services Architecture (OGSA) extends the Web Services terminology to include Grid concepts, and to manage the creation and termination of resources as services. Its main focus is on the definition of abstract interfaces that allow services to cooperate without too much concern about the actual protocols being used. OGSA is an architecture which is under constant development, changing continuously in response to feedback from the user community. Grid Services, as defined by OGSA, integrate Grid technologies from Globus toolkits with Web Services mechanisms to construct a Grid-based distributed framework. A Grid Service instance is a potentially transient service that conforms to a set of conventions, expressed as Web Service Description Language (WSDL) [23] interfaces, extensions, and behaviors. Grid Services provide controlled management of the distributed and often long-lived state that is commonly required in sophisticated distributed applications.

The release of the Globus Toolkit 4 (GT4) provides a simple approach to implementing OGSA. Instead of adding web service functionality to existing Grid middleware, as done before in previous versions, web services are now used as a fundamental core for the Grid middleware with a specification called the Web Service Resource Framework (WSRF) [20]. The WSRF is a set of six Web services specifications that define the Web Service Resource approach to modeling and managing state in a Web services context.

3 Production Grids

3.1 The Large Hadron Collider Computing Grid and CrossGrid

The Large Hadron Collider (LHC), which is currently being built at CERN, is a scientific instrument capable of producing millions of Gigabytes of scientific data. In order to allow thousands of scientists to analyse this data, a data storage and analysis infrastructure capable of handling such amounts of information is to be developed. The current production middleware used in this project is called Large Hadron Collider Computing Grid (LCG). It is based on previous efforts from the European DataGrid project and the Globus Toolkit 2 release. The LCG middleware is one of the most popular production Grid middleware used today, with over 100 sites in more than 30 different countries participating with the LCG project. For our biomedical experiments on the Grid, we rely on the CrossGrid testbed, which is built on the LCG middleware. The testbed includes a local step, typically inside a research center or university via Fast or Gigabit Ethernet, a jump via a national network provider at speeds that will range from 34Mbits/s to 622Mbits/s or even Gigabit, and a link to the Geant European network at 155 Mbits/s to 2.5 Gbits/s.

3.2 The VRE Parallel Biomedical Application on the Grid

For our interoperability experiments we use a parallel solver from the Virtual Radiology Explorer (VRE), a Grid-based problem solving environment developed by the University of Amsterdam. We deployed our problem solving environment within the European Crossgrid project [26], laying out our system architecture using the Grid as a medium, with a validated case study in vascular reconstruction. The biomedical application at the core consists of a parallel computational hemodynamics solver [27] that computes pressure, velocities, and shear stresses during a full systolic cycle. The data used as input for simulation can be obtained from several imaging techniques used to detect vascular disorders; for instance, 3D data acquired by Computed Tomography or Magnetic Resonance Imaging, or particularly Magnetic Resonance Angiography for imaging blood vessels that contain flowing blood.

4 Grid Interoperability Analysis

4.1 Data Grid Analysis

One of the main goals in our distributed biomedical project is the development and maintenace of a next generation production system that scales well while our Grid testbed grows and our virtual organization is expanded with OGSA resources. Our basic security requirements are covered by the Grid Security Infrastructure (GSI) and its public key services, so our priority is to make sure that our biomedical data access from Storage Elements (SEs) remains reliable, and that the data management services do not overwhelm our resources. For our data-centered experiments, our focus is on enabling our users transparent data access for analysis and computational job submission, acccording to a basic usage scenario. The user of our application loads the biomedical medical data, selects a region of interest, and submits the analized data for simulation. After the job has been completed, the results are transferred to the local machine for visualization and rendering. This scenario assumes a number of data transfers, both input segmented data for the simulation kernel, as well as simulation output to be visualized. In our current LCG-based infrastructure we use Globus GridFTP and DataGrid Replica Locator Service (RLS) to transfer our data between Grid nodes. We set out to replicate some characteristics of this behaviour between our current testbed and a few newly added GT4 nodes. We tranfered a number of representative compressed geometry input datasets between new GT4 and CrossGrid LCG SEs spread out across Europe, measuring transfer times for comparing performance of the GridFTP implementations, as shown in Fig. 1.

Once we successfully transfered the input geometry data files between GT4, LCG2.0, LCG2.3 and LCG 2.6 resources, we then made a number of transfers using larger simulation output files, studying CPU usage to get an idea of the overhead caused by the transfers. Fig. 1 shows some of our results, e.g., that the transfer times for the input geometries were consistently below 2 seconds and below 30 seconds for the simulation output (taking more time due to network

file transfer performance	small.bs (16kb)	large.bs(107kb)	velocity.bs (43392kb)
hilde-ridgrid2 ave	0.64	0.61	4.51
ridgrid2-hilde ave	0.65	0.68	5.59
se010.fzk.de-ridgrid2 ave	1.45	1.55	26.74
ridgrid2-se010.fzk.de ave	0.85	1.07	26.55
ds2d-ridgrid2 ave	0.49	0.51	4.33
ridgrid2-ds2d	X	X	X

Fig. 1: Overview of file transfer performance between different machines in our experimental testbed, showing average transfer times in seconds between hilde (LCG2.6) to ridgrid2(GT4), se010.fzk.de (LCG2.3) and ridgrid2, and ds2d (LCG2.0) and ridgrid2; note that transfer from GT4 to LCG2.0 does not work due to protocol issues.

latency between Dutch and German sites). Transfers from GT4 machines to LCG 2.0 CG production machines failed due to protocol incompatibilities.

4.2 Computational Grid Analysis

The recent addition of new resources to our Grid infrastructure continuously provides us with potential increased computational capacity. However, continual evolution in Grid middleware raises issues of structural interoperability, specially when migrating from established architectural paradigms to new ones. The main focus of this computation-centered experiments is on finding means to make use of these new resources for job submission, across Grid middleware architectures. In our current Grid infrastructure, which relies on the CrossGrid middleware, we make use of both MPICH-P4 and MPICH-G2 MPI devices to run our application. Although support for MPICH-P4 is available for more recent Grid production middleware releases, MPICH-G2 support is usually known to be unstable, at best. In order to explore the support for both MPICH-P4 and MPICH-G2, we proceeded to execute and measure the performance of the application on three different platforms. These three platforms include our current infrastructure (LCG2.0 for production and LCG 2.3 for development versions), the latest LCG 2.6 production middleware release of the EGEE project, and the Globus Toolkit 4. We used globus-job-run and globusrun-ws, where available, for our MPICH-P4 runs, while MPICH-G2 runs were initiated directly by Globus Resource Specification Language.

5 Discussion and Conclusions

During our experiments we found that, given the relatively immaturity of the core transfer utilities in GT4, job submission and data management services running on the SEs used for our experiments were quite stable, even though the resources are shared across different middleware architectures. Our experiments showed fairly good interoperability for data access, except when using a GT4 client accessing a LCG2.0 gsiftp server, for reasons discussed earlier. Data

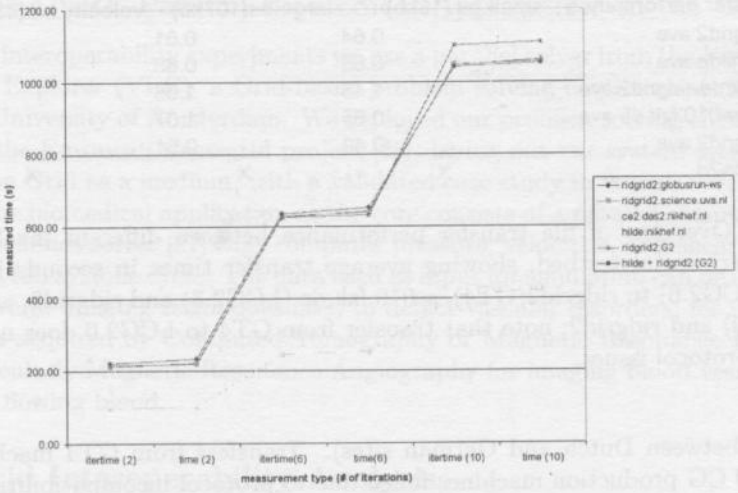

Fig. 2: Total execution time (time measured from submission to completion) and iterating time (time per iteration*number of iterations) in seconds for 2, 6 and 10 simulation time steps). The LCG2.0 machine (ce2) performs significantly faster due to superior hardware specifications. The iterating time of Hilde(LCG 2.6) is slightly lower than that of ridgrid2(GT4), but the total execution time tends to be equal or slightly higher. Entries marked with "G2" are job runs performed with MPICH-G2. The ridgrid2:globusrun-ws indicates a run using the GT4 WS-based job submission tool. All other entries are runs using globus-job-run. All the runs were made using 2 processes.

transfer times were not noticeably different between toolkits, with globusrun-ws showing about 3 seconds of overhead over globus-job-run; the overhead for LCG 2.6 due to scheduling policies of the job manager, where the queue is only checked periodically. Also, GT4 interoperated surprisingly well for job submission, and across site submissions using MPICH-G2, which performance proved comparable to MPICH-P4, to our surprise. Job submissions run well on all of the three tested platforms for MPICH-P4. We were pleasantly surprised by our relatively smooth experience running MPICH-G2 simultaneously on both the LCG 2.6 and GT4 middleware, with runs performed on both platforms and providing a competitive total execution time, regardless of the location where the run was initiated. To the best of our knowledge, this is the first work to report successful runs of a real computationally intensive application using multiple nodes featuring different middleware in a production-type testbed.

6 Future Work

For future work we plan to perform wider-scale experiments including the intensive use of RLS, dynamically staging of large output datasets using the MPICH-

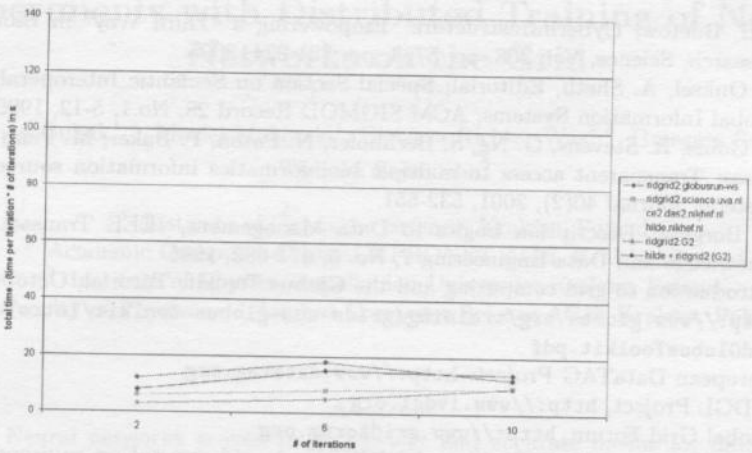

Fig. 3: Detail of job submission performance shown in Fig 2. Iterating time is subtracted from total execution time in order to measure scheduling and initialization overhead. Executing globusrun-ws instead of globus-job-run on ridgrid2 introduces an additional overhead of approximately 3 seconds. The Hilde machine displays a much higher and variable overhead.

G2 device in order to get a better insight into issues of cross-site interoperability and performance overhead associated with such a hybrid approach to Grid-based biomedical data access. We also plan to look into the ability of staging files that serve as input for the BStream job, and the capacity to provide and manage sandboxes for both job input and output. Future releases of EGEE project middleware, such as Glite, provide yet another new platform to examine. Examining these future production releases is also considered to be future work. Finally, we plan to perform more extensive and intensive runs so that we can present a more accurate view of the exact performance of our Biomedical PSE on our testing sites, particularly with new implementations of the WSRF.net architecture.

Acknowledgements. The authors wish to acknowledge the CrossGrid team, Lilit Abrahamyan, and Dennis Kaarsemaker for their support and help.

References

1. I. Foster, C. Kesselmann, and S. Tuecke; The anatomy of the grid: Enabling scalable virtual organizations, International Journal of Supercomputer Applications, vol. 15, no. 3, 2001
2. Breton V., Medina R. and Montagnat J. 2003; DataGrid, prototype of a biomedical grid Methods Inf. Med. 42, 143-7
3. CrossGrid, `http://www.crossgrid.org`
4. M. Rambadt, P. Wieder; UNICORE – Globus Interoperability: Getting the Best of Both Worlds, HPDC, 2002

5. K.H. Buetow; Cyberinfrastructure: Empowering a "Third Way" in Biomedical Research, Science, Vol. 308. no. 5723, pp. 821-824, 2005
6. A. Ouksel, A. Sheth, Editorial; Special Section on Semantic Interoperability in Global Information Systems, ACM SIGMOD Record 28, No.1, 5-12, 1999
7. C. Goble, R. Stevens, G. Ng, S. Bechhofer, N. Paton, P. Baker, M. Peim and A. Brass; Transparent access to multiple bioinformatics information sources. IBM Systems Journal 40(2), 2001, 532-551
8. A. Borgida; Description Logics in Data Management, IEEE Transactions on Knowledge and Data Engineering 7, No. 5, 671-682, 1995
9. Introduction to grid computing and the Globus Toolkit. Tutorial, October 2001. http://www.globus.org/training/grids-and-globus-toolkit/IntroToGrids-AndGlobusToolkit.pdf
10. European DataTAG Project: http://www.datatag.org
11. iVDGL Project, http://www.ivdgl.org
12. Global Grid Forum, http://www.gridforum.org
13. D.W. Erwin and D.F. Snelling; UNICORE: A grid computing environment, in Proceedings of Euro-Par 2001, pp. 825-834, Springer LNCS 2150, August 2001.
14. F. Donno, V. Ciaschini, D. Rebatto, L. Vaccarossa, M. Verlato; The WorldGrid transatlantic testbed: a successful example of Grid interoperability across EU and US domains. Proceedings of the Conference on Computing in High-Energy Physics, La Jolla CA, USA, March 24-28, 2003.
15. GRIP, http://www.grid-interoperability.org
16. Ricardo Manuel Brito da Rocha; Evaluation of Emerging Grid Technologies OGSI and the Globus Toolkit 3, CERN Genebra. FEUP Porto, Relatorio do Estagio Curricular da LEIC 2002/2003
17. Foster, I., Kesselman, C., Nick, J.M. and Tuecke, S.; The Physiology of the Grid. Computer, 35(6), 2002
18. Kreger, H. 2001; Web Services Conceptual Architecture (WSCA 1.0). http://www-4.ibm.com/software/solutions/webservices/.
19. I. Foster; Globus Toolkit Version 4: Software for Service-Oriented Systems, IFIP International Conference on Network and Parallel Computing, 2005
20. Foster, I., J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, T. Storey, and S. Weerawaranna; Modeling Stateful Resources with Web Services. 2004, Globus Alliance.
21. The LHC Computing Grid, http://lcg.web.cern.ch
22. The European Data Grid, http://www.eu-datagrid.org
23. R. Chinnici, M. Gudgin, J.J. Moreau, J. Schlimmer; Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Working Draft. http://www.w3.org/TR/2004/WD-wsdl20, 2004
24. B. Sotomayor; The Globus Toolkit 4 Programmer's Tutorial, http://gdp.globus.org/gt4-tutorial/singlehtml/progtutorial0.2.html
25. F. Carminati; GEANT Users Guide, CERN Program Library, 1991 (unpublished).
26. A. Tirado-Ramos, P.M.A. Sloot, A.G. Hoekstra and M. Bubak; An Integrative Approach to High-Performance Biomedical Problem Solving Environments on the Grid, Parallel Computing, (special issue on High-Performance Parallel Biocomputing) vol. 30, nr 9-10 pp. 1037-1055. (Chun-Hsi Huang and Sanguthevar Rajasekaran, editors), 2004.
27. Simulation of a systolic cycle in a realistic artery with the Lattice Boltzmann BGK method, A.M. Artoli, A.G. Hoekstra, P.M.A. Sloot, Int. J. Mod. Phys. B, 2003.