Execution of a Bioinformatics Application in a Joint IRISGrid/EGEE Testbed*

J. L. Vázquez-Poletti[1], E. Huedo[2], R. S. Montero[1], and I. M. Llorente[1,2]

[1] Departamento de Arquitectura de Computadores y Automática. Facultad de Informática, Universidad Complutense de Madrid. 28040 Madrid, Spain.
[2] Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas. Centro de Astrobiología (CSIC-INTA). 28850 Torrejón de Ardoz, Spain.

Abstract. This paper describes the execution of a Bioinformatics application over the largest ever spanish testbed. This testbed is composed of resources devoted to EGEE and IRISGrid projects and has been integrated by taking advance of the modular, decentralized and "end-to-end" architecture of the Grid*W*ay framework. Results show the feasibility of building loosely-coupled Grid environments only based on Globus services, while obtaining non trivial levels of quality of service. Such approach allows a straightforward resource sharing as the resources are accessed by using *de facto* standard protocols and interfaces.

1 Introduction

Different Grid infrastructures are being deployed within growing national and transnational research projects. The final goal of these projects is to provide the end user with much higher performance than that achievable on any single site. However, from our point of view, it is arguable that some of these projects embrace the Grid philosophy, and to what extent. This philosophy, proposed by Foster [1], claims that *a Grid* is a system (i) not subject to a centralized control and (ii) based on standard, open and general-purpose interfaces and protocols, (iii) while providing some level of quality of service (QoS), in terms of security, throughput, response time or the coordinated use of different resource types. In current projects, there is a tendency to ignore the two first requirements in order to get higher levels of QoS. However, those requirements are even more important because they are the key to the success of *the Grid*.

The Grid philosophy leads to computational environments, which we call *loosely-coupled Grids*, mainly characterized by [2]: autonomy (of the multiple administration domains), heterogeneity, scalability and dynamism. In a loosely-coupled Grid, the different layers of the infrastructure should be separated and

* This research was supported by Ministerio de Ciencia y Tecnología, through the research grants TIC 2003-01321 and 2002-12422-E, and by Instituto Nacional de Técnica Aeroespacial "Esteban Terradas" (INTA) – Centro de Astrobiología. The authors participate in the EGEE project, funded by the European Union under contract IST-2003-508833.

independent, being communicated between them with a limited and well defined set of interfaces and protocols. This layers are [2]: Grid fabric, core Grid middleware, user-level Grid middleware, and Grid applications.

The coexistence of several projects, each with its own middleware developments, adaptations or extensions, arise the idea of using them simultaneously (from an user's viewpoint) or contribute the same resources to more than one project (from an administrator's viewpoint). One approach could be the development of gateways between different middleware implementations [3]. Other approach, more in line with the Grid philosophy, is the development of client tools that can adapt to different middleware implementations. We hope this could lead to a shift of functionality from resources to brokers or clients, allowing the resources to be accessed in a standard way and easing the task of sharing resources between organizations and projects. We should consider that the Grid not only involves the technical challenge of constructing and deploying this vast infrastructure, it also brings up other issues related to security and resource sharing policies [4] as well as other socio-political difficulties [5].

Practically, the majority of the Grid infrastructures are being built on protocols and services provided by the Globus Toolkit[1], becoming a *de facto* standard in Grid computing. Globus architecture follows an hourglass approach, which is indeed an "end-to-end" principle [6]. Therefore, instead of succumbing to the temptation of tailoring the core Grid middleware to our needs (since in such case the resulting infrastructure would be application specific), or homogenizing the underlying resources (since in such case the resulting infrastructure would be a highly distributed cluster), we propose to strictly follow the "end-to-end" principle. Clients should have access to a wide range of resources provided through a limited and standardized set of protocols and interfaces. In the Grid, these are provided by the core Grid middleware, Globus, just as, in the Internet, they are provided through the TCP/IP set of protocols. Moreover, the "end-to-end" principle reduces the firewall configuration to the minimum, which is also welcome by the security administrators.

One of the most ambitious projects to date is EGEE[2] (Enabling Grids for E-sciencE), which is creating a production-level Grid infrastructure providing a level of performance and reliability never achieved before. EGEE currently uses the LCG (LHC Computing Grid)[3] middleware, which is based on Globus. Other much more modest project is IRISGrid[4] (the Spanish Grid Initiative), whose main objective is the creation of a stable national Grid infrastructure. The first version of the IRISGrid testbed is based only on Globus services, and it has been widely used through the Grid*W*ay framework[5].

For the purposes of this paper we have used a Globus-based testbed to run a Bioinformatics application through the Grid*W*ay framework. This testbed was

[1] http://www.globus.org
[2] http://www.eu-egee.org
[3] http://lcg.web.cern.ch
[4] http://irisgrid.rediris.es
[5] http://www.gridway.org

built up from resources inside IRISGrid and EGEE projects. The aim of this paper is to demonstrate the application of an "end-to-end" principle in a Grid infrastructure, and the feasibility of building loosely-coupled Grid environments only based on Globus services, while obtaining non trivial levels of quality of service through an appropriate user-level Grid middleware.

The structure of the paper follows the layered structure of Grid systems. The Grid fabric is described Section 2. Section 3 describes the core Grid middleware. Section 4 introduces the functionality and characteristics of the Grid*W*ay framework, used as user-level Grid middleware. Section 5 describes the target application. Finally, Section 6 presents the obtained results and Section 7 ends up with some conclusions.

2 Grid Fabric: IRISGrid and EGEE resources

This work has been possible thanks to the collaboration of those research centers and universities that temporarily shared some of their resources in order to set up a geographically distributed testbed. The testbed results in a very heterogeneous infrastructure, since it presents several middlewares, architectures, processor speeds, resource managers (RM), network links, etc. A brief description of the participating resources is shown in Table 1.

Some centers are inside IRISGrid, which is composed of around 40 groups from different spanish institutions. In the experiment, 7 of them participated donating a total number of 195 CPUs. Other centers participate in the EGEE project, which is composed of more than 100 contracting and non-contracting partners. In the experiment, 7 spanish centers participated, donating a total number of 333 CPUs.

Together, the testbed is composed of 13 sites (note that LCASAT-CAB is both in IRISGrid and EGEE) and 528 CPUs. In the experiments below, we limited to four the number of jobs simultaneously submitted to the same resource, with the aim of not saturating the whole testbed, so only 64 CPUs were used at the same time. All sites are connected by RedIRIS, the Spanish Research and Academic Network. The geographical location and interconnection links of the different sites are shown in Figure 1.

3 Core Grid Middleware: Globus

Globus services allow secure and transparent access to resources across multiple administrative domains, and serve as building blocks to implement the stages of Grid scheduling [7]. Table 2 summarizes the core Grid middleware components existing in both IRISGrid and EGEE resources used in the experiments. In the case of EGEE, we only use Globus basic services, ignoring any higher-level services, like the resource broker or the replica location service.

We had to introduce some changes in the security infrastructure in order to perform the experiments. For authentication, we used a user certificate issued by DATAGRID-ES CA, so we had to give trust to this CA on IRISGrid

Table 1. IRISGrid and EGEE resources contributed to the experiment.

Testbed	Site	Resource	Processor	Speed	Nodes	RM
IRISGrid	RedIRIS	heraclito	Intel Celeron	700MHz	1	Fork
		platon	2$\times$Intel PIII	1.4GHz	1	Fork
		descartes	Intel P4	2.6GHz	1	Fork
		socrates	Intel P4	2.6GHz	1	Fork
	DACYA-UCM	aquila	Intel PIII	700MHz	1	Fork
		cepheus	Intel PIII	600MHz	1	Fork
		cygnus	Intel P4	2.5GHz	1	Fork
		hydrus	Intel P4	2.5GHz	1	Fork
	LCASAT-CAB	babieca	Alpha EV67	450MHz	30	PBS
	CESGA	bw	Intel P4	3.2GHz	80	PBS
	IMEDEA	llucalcari	AMD Athlon	800MHz	4	PBS
	DIF-UM	augusto	4$\times$Intel Xeon**	2.4GHz	1	Fork
		caligula	4$\times$Intel Xeon**	2.4GHz	1	Fork
		claudio	4$\times$Intel Xeon**	2.4GHz	1	Fork
	BIFI-UNIZAR	lxsrv1	Intel P4	3.2GHz	50	SGE
EGEE	LCASAT-CAB	ce00	Intel P4	2.8GHz	8	PBS
	CNB	mallarme	2$\times$Intel Xeon	2.0GHz	8	PBS
	CIEMAT	lcg02	Intel P4	2.8GHz	6	PBS
	FT-UAM	grid003	Intel P4	2.6GHz	49	PBS
	IFCA	gtbcg12	2$\times$Intel PIII	1.3GHz	34	PBS
	IFIC	lcg2ce	AMD Athlon	1.2GHz	117	PBS
	PIC	lcgce02	Intel P4	2.8GHz	69	PBS

** These resources actually present two physical CPUs but they appear as four logical CPUs due to hyper-threading

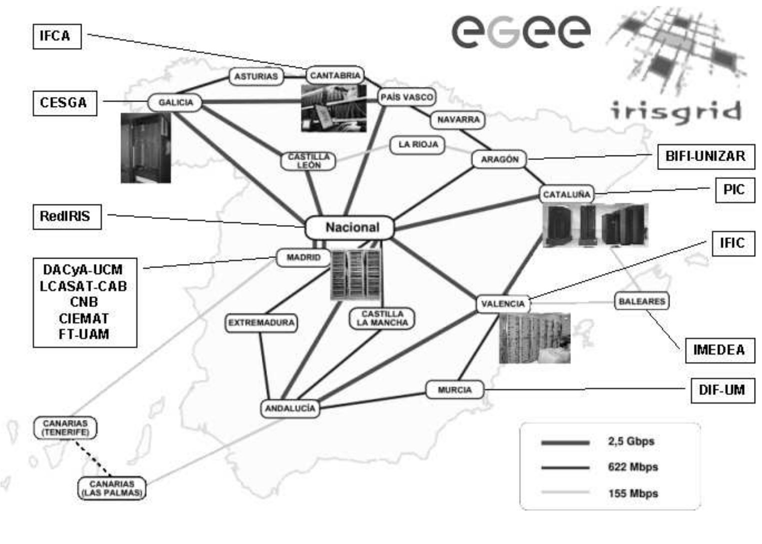

Fig. 1. Geographical distribution and interconnection network of sites.

Table 2. Core Grid middleware.

Component	IRISGrid	EGEE
Security Infrastructure	IRISGrid CA and manually generated `grid-mapfile`	DATAGRID-ES CA and automatically generated `grid-mapfile`
Resource Management	GRAM with shared home directory in clusters	GRAM without shared home directory in clusters
Information Services	IRISGrid GIIS and local GRIS, using the MDS schema	CERN BDII and local GRIS, using the GLUE schema
Data Management	GASS and GridFTP	GASS and GridFTP

resources. Regarding authorization, we had to add an entry for the user in the `grid-mapfile` in both IRISGrid and EGEE resources. Another possibility would be to use two different user certificates, each to access one testbed. Moreover, in large projects there are VO management systems, like VOMS, so it could be possible to create gateways between them, so we can have a VO in EGEE consisting of all the IRISGrid users and viceversa [3].

4 User-Level Grid Middleware: Grid*W*ay

User-level middleware is required in the client side to make it easier and more efficient the execution of applications. Such client middleware should provide the end user with portable programming paradigms and common interfaces.

In a Globus-based environment, the user is responsible for manually performing all the submission steps [7] in order to achieve any functionality. To overcome this limitation, Grid*W*ay [8] was designed with a *submit & forget* philosophy in mind. The core of the Grid*W*ay framework is a personal *submission agent* that performs all scheduling stages and watches over the correct and efficient execution of jobs on Globus-based Grids. The Grid*W*ay framework provides adaptive scheduling and execution, as well as fault tolerance capabilities to handle the dynamic Grid characteristics.

A key aspect in order to follow the "end-to-end" principle is how job execution is performed. In EGEE, file transfers are initiated by a job wrapper running in the compute nodes, therefore they act as client machines, so needing network connectivity and client tools to interact with the middleware. In Grid*W*ay, however, job execution is performed in three steps by the following modules:

1. *prolog*: It prepares the remote system by creating a experiment directory and transferring the input files from the client.
2. *wrapper*: It executes the actual job and obtains its exit status code.

3. *epilog*: It finalizes the remote system by transferring the output files back to the client and cleaning up the experiment directory.

This way, Grid*W*ay doesn't rely on the underlying middleware to perform preparation and finalization tasks. Moreover, since both *prolog* and *epilog* are submitted to the front-end node of a cluster and *wrapper* is submitted to a compute node, Grid*W*ay doesn't require any middleware installation nor network connectivity in the compute nodes.

Other projects [9–12] have also addressed resource selection, data management, and execution adaptation. We do not claim innovation in these areas, but remark the advantages of our modular, decentralized and "end-to-end" architecture for job adaptation to a dynamic environment.

In this case, we have taken full advance of the modular architecture of Grid*W*ay, as we didn't have to directly modify the source code of the *submission agent*. Instead, we had to only modify several modules that were implemented as scripts, so the modification was straightforward. For example, another *resource selector* module had to be developed in order to understand the GLUE schema used in EGEE, since the former one only understood the Globus MDS schema. We also developed a *super resource selector* to merge results from the other two. The *wrapper* module had also to be modified in order to perform an explicit file staging between the front-end and the compute nodes in EGEE clusters.

5 Grid Application: Computational Proteomics

In the following experiments, we will consider a Bioinformatics application aimed at predicting the structure and thermodynamic properties of a target protein from its amino acid sequence. The algorithm, tested in the 5th round of Critical Assessment of techniques for protein Structure Prediction (CASP5)[6], aligns with gaps the target sequence with all the 6150 non-redundant structures in the Protein Data Bank (PDB)[7], and evaluates the match between sequence and structure based on a simplified free energy function plus a gap penalty item. The lowest scoring alignment found is regarded as the prediction if it satisfies some quality requirements. In such cases, the algorithm can be used to estimate thermodynamic parameters of the target sequence, such as the folding free energy and the normalized energy gap [13].

We have applied the algorithm to the prediction of thermodynamic properties of families of orthologous proteins, i.e. proteins performing the same function in different organisms. The biological results of the comparative study of several families of orthologous proteins are presented elsewhere [14].

6 Experiences and Results

The experiments presented here consist in the analysis of a family of 80 orthologous proteins of the *Triose Phosphate Isomerase* enzyme (an enzyme is a special

[6] http://PredictionCenter.llnl.gov/casp5/
[7] http://www.pdb.org

case of protein). Five experiments were conducted in different days during a week. The average turnaround time for the five experiments was 43.37 minutes.

Figure 2 shows the dynamic throughput achieved during the five experiments alongside the theoretical throughput of the most powerful site, where the problem could be solved in the lowest time, in this case DIF-UM (taking into account the limitation in the number of simultaneously used nodes). The throughput achieved on each experiment varies considerably, due to the dynamic availability and load of the testbed. For example, resource ce00 at site LCASAT-CAB was not available during the execution of the first experiment. Moreover, fluctuations in the load of network links and computational resources induced by non-Grid users affected to a lesser extent in the second experiment, as it was performed at midnight.

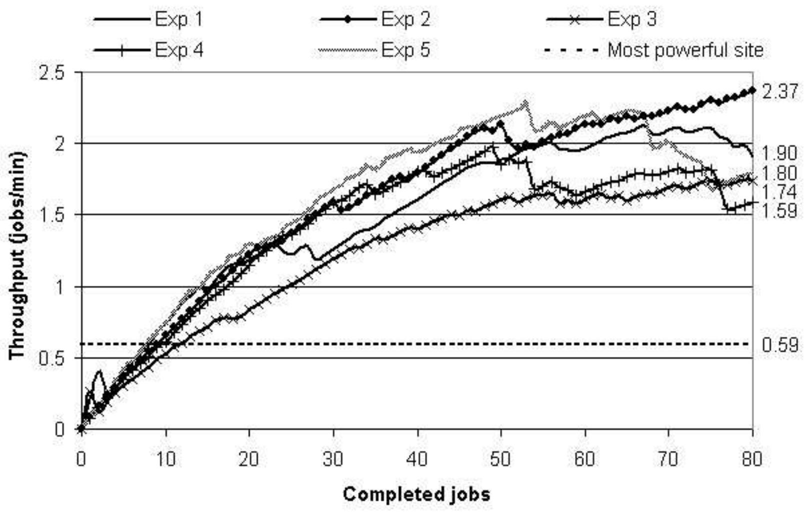

Fig. 2. Testbed dynamic throughput during the five experiments and theoretical throughput of the most powerful site.

7 Conclusions

We have shown that the "end-to-end" principle works at the client side (i.e. the user-level Grid middleware) of a Grid infrastructure. Our proposed user-level Grid middleware, Grid*W*ay, can work with Globus, as a standard core Grid middleware, over any Grid fabric in a *loosely-coupled* way. The easy process of integration of two so different testbeds, although both are based on Globus, demonstrates that the Grid*W*ay approach (i.e. the Grid way), based on a modular, decentralized and "end-to-end" architecture, is appropriate for the Grid.

Moreover, loosely-coupled Grids allow a straightforward resource sharing since resources are accessed and exploited through *de facto* standard protocols and interfaces, similar to the early stages of the Internet. This way, the loosely-coupled model allows an easier, scalable and compatible deployment.

8 Acknowledgments

We would like to thank all the institutions involved in the IRISGrid initiative and the EGEE project, in particular those who collaborated in the experiments. We would like to also thank Ugo Bastolla, staff scientist in the Bioinformatics Unit at Centro de Astrobiología (CAB) and developer of the Bioinformatics application used in the experiments.

References

1. Foster, I.: What Is the Grid? A Three Point Checklist. GRIDtoday **1** (2002) Available at http://www.gridtoday.com/02/0722/100136.html.
2. Baker, M., Buyya, R., Laforenza, D.: Grids and Grid Technologies for Wide-Area Distributed Computing. Software – Practice and Experience **32** (2002) 1437–1466
3. Allan, R.J., Gordon, J., McNab, A., Newhouse, S., Parker, M.: Building Overlapping Grids. Technical report, University of Cambridge (2003)
4. San José, O., Suárez, L.M., Huedo, E., Montero, R.S., Llorente, I.M.: Resource Performance Management on Computational Grids. In: Proc. 2nd Intl. Symp. Parallel and Distributed Computing (ISPDC 2003), IEEE CS (2003) 215–221
5. Schopf, J.M., Nitzberg, B.: Grids: The Top Ten Questions. Scientific Programming, special issue on Grid Computing **10** (2002) 103–111
6. B. Carpenter, E.: RFC 1958: Architectural Principles of the Internet (1996)
7. Schopf, J.M.: Ten Actions when Superscheduling. Technical Report GFD-I.4, Scheduling Working Group – The Global Grid Forum (2001)
8. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Intl. J. Software – Practice and Experience (SPE) **34** (2004) 631–651
9. Berman, F., Wolski, R., Casanova, H., et al.: Adaptive Computing on the Grid Using AppLeS. IEEE Trans. Parallel and Distributed Systems **14** (2003) 369–382
10. Buyya, R., D.Abramson, Giddy, J.: A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. Future Generation Computer Systems **18** (2002) 1061–1074
11. Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor/G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing **5** (2002) 237–246
12. Vadhiyar, S., Dongarra, J.: A Performance Oriented Migration Framework for the Grid. In: Proc. 3rd Intl. Symp. Cluster Computing and the Grid (CCGrid 2003), IEEE CS (2003) 130–137
13. Bastolla, U., Vendruscolo, M., Knapp, E.W.: A Statistical Mechanical Method to Optimize Energy Parameters for Protein Folding. Proc. National Academy of Sciences (PNAS) of USA **97** (2000) 3977–3981
14. van Ham, R., Kamerbeek, J., Palacios, C., et al.: Reductive Genome Evolution in Buchnera Aphidicola. Proc. National Academy of Sciences (PNAS) of USA **100** (2003) 581–586