A Comparison Between two Grid Scheduling Philosophies: EGEE WMS and Grid *W*ay *

J. L. Vázquez-Poletti†
E. Huedo
R. S. Montero
I. M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid
28040 Madrid, Spain

May 31, 2007

Abstract

In order to achieve a reasonable degree of performance and reliability, Metascheduling has been revealed as a key functionality of the grid middleware. The aim of this paper is to provide a comparative analysis between two major grid scheduling philosophies: a semi-centralized approach, represented by the EGEE Workload Management System, and a fully distributed approach, represented by the Grid *W*ay Metascheduler. The distributed approach follow a loosely-coupled philosophy for the Grid resembling the *end-to-end* principle, which has fostered the spectacular development and diffusion of the Internet and, in particular, Web technologies in the past decade. The comparative is both theoretical, through a functionality checklist, and experimental, through the execution of a fusion physics plasma application on the EGEE infrastructure. This paper not only includes a standard analysis with the obtained times, but also a complex analysis based on a performance model.

Keywords: Grid Computing, Metascheduling, EGEE WMS, Grid *W*ay, Fusion Physics.

*This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through the research grant TIN2006-2806.

†Corresponding author. Phone: +34 91 394 75 41. Fax: +34 91 394 46 87. E-mail: jlvazquez@fdi.ucm.es

1 Introduction

The growing computational needs of nowadays projects have permitted the evolution to a new paradigm called Grid Computing. The ability to have applications draw computing power from a global resource pool to achieve high performance has become a new challenge for distributed-computing and Internet technologies. Several research centers share their computing assets in grids, which dramatically increase the number of processing and storage resources applications can access.

Different grid infrastructures are being deployed in the context of growing national and international research projects. Among the intervening elements of a computional grid, the Metascheduler is gathering most attention as a way to meet the challenging needs of several application domains. The term Metascheduler can be defined as a grid middleware that discovers, evaluates and allocates resources for grid jobs by coordinating activities between multiple heterogeneous schedulers that operate at local or cluster level [1]. In general, the scheduling process includes the following phases: resource discovery and selection; and job preparation, submission, monitoring, migration and termination [2].

Although we can find several philosophies for the Grid, discussed in Section 2, as well as different implementations of the Metascheduler, some of them enumerated in Section 3, the objective of this paper is to compare Grid*Way* [3] and the EGEE Workload Management System (WMS) as two representative implementations. A short overview of their architectures can be found in Section 4 along with a comparative analysis of the functionality provided by both solutions. Then, Section 5 evaluates the fine-grain metrics and performance obtained by the two alternatives in the execution of a fusion physics plasma application on the EGEE infrastructure. Finally, the paper ends up with some conclusions and future work in Section 6.

2 A Loosely-coupled Philosophy for the Grid

A Grid infrastructure is usually decomposed into the following layers [4]: Grid applications and portals; user-level Grid middleware; core Grid middleware; and Grid fabric. The two internal layers are called *the middleware*, since they connect applications with resources, or Grid fabric. In a *loosely-coupled* Grid, it is important to keep these layers separated and independent with a limited and well defined set of interfaces and protocols between them (Figure 1).

The main characteristics of these environments are [4] autonomy (of the multiple administration domains), heterogeneity, scalability and dynamism. These properties completely determine the way that scheduling and execution on Grids have to be done. For example, scalability and autonomy prevent the deployment of centralized resource brokers with total control over client requests and resource status. On the other hand, the dynamic resource characteristics in terms of availability, capacity and cost make essential the ability to adapt job scheduling and execution to these conditions.

The Globus Toolkit [5] has become a de facto standard core middleware in Grid Computing. 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 [2]. Globus architecture follows an hourglass model, which is indeed an *end-to-end* principle [6]. Therefore, instead of succumbing to the temptation of tailoring the core Grid middleware to its needs (since in such case the resulting infrastructure would be a highly distributed cluster), the *loosely-coupled* philosophy follows 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, i.e. Globus. Just as, in the Internet, they are provided through the IP protocol.

The application of the *end-to-end* principle to Grid computing requires user-level middleware, such as Grid*W*ay, 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, which are being standardized by the Open Grid Forum (OGF), see, for example the Distributed Resource Management Application API (DRMAA) [7] and the Simplified API for Grid Application (SAGA) [8].

The EGEE project is creating the largest production-level Grid Infrastructure in the world, which provides a level of performance and reliability never achieved before. A very restrictive set of requirements has been established for organizations that wish to take part in it. EGEE defines the user-level Grid middleware, the core Grid middleware and the Grid fabric, and so these layers are tightly related. EGEE uses the LCG (LHC Computing Grid)[1] middleware, LCG-2.

This architecture presents some limitations in terms of heterogeneity, as it has fixed configuration for clusters. The scalability of its deployment is also limited, as the middleware should be installed on the compute nodes, and they should have network connectivity. Also, LCG's focus is mainly on particle physics applications and its development is highly dependant on a single organization, CERN (the European Organization for Nuclear Research). Nevertheless, it is expected that the new EGEE middleware, gLite[2], will overcome to some extent some of these limitations.

3 Related Work

Among the several implementations of the Metascheduler, the authors found representative CSF, Condor-G, VIOLA MetaScheduling Service and Gridbus Service Broker. CSF (Community Scheduler Framework) [9] is a WSRF-compliant Metascheduler built upon the Globus Toolkit. Depending if the access to a specific GRAM adapter is desired, the installation of LSF[3] is needed. It sup-

[1] http://lcg.web.cern.ch/
[2] http://www.glite.org/
[3] http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/

ports advance reservation booking and offers round-robin and reservation based scheduling algorithms.

Condor-G [10] is not conceived for supporting scheduling policies, but on the other hand, it supplies mechanisms that may be useful for a Metascheduler standing above, like *ClassAd* and *DAGMan*. *ClassAd* [11] (Classified Advertisement Language) allows the user to decide the resources for the job in the matchmaking process. *DAGMan* [12] (Directed Acyclic Graph Manager) is a workflow manager where interdependencies between jobs or data can be specified.

VIOLA [13] (Vertically Integrated Optical Testbed for Large Applications) is a project which aims to take benefit of an optical testbed using advanced techniques and services[4]. Its MetaScheduling Service stands on UNICORE [14] and provides support for complex distributed applications, as well as automation of co-allocation of computational and network resources. Web-services allow to either implement adapters for local resource managers, and access the MetaScheduling Service itself.

Gridbus Service Broker [15] accesses transparently to various low level middleware solutions like Globus Toolkit, XGrid [16], UNICORE and Alchemi [17]. Its scheduling strategies consider both job deadline and budget specified by the user, in order to optimize the resource usage. On the other hand, custom scheduling algorithms can be implemented by means of writing a custom scheduler.

4 Grid Scheduling Infrastructures

The EGEE WMS and Grid*Way* are representative metascheduling technologies of different strategies to deploy a grid scheduling infrastructure. Grid*Way* follows the *loosely-coupled* Grid principle described in Section 2, mainly characterized by: autonomy of the multiple administration domains, heterogeneity, scalability and dynamism. A Grid*Way* instance is installed in each organization involved in the partner grid to provide scheduling functionality for intra-organization users (site-level scheduling). On the other hand, EGEE WMS provides a higher centralized strategy as one or more scheduling instances can be installed in the grid infrastructure, each providing scheduling functionality for a number of VOs (VO-level scheduling).

The EGEE WMS is the result of previous projects (Datagrid, CrossGrid). The present version of the EGEE WMS is based on the LCG-2 middleware and it is migrating to a new one called gLite which inherits many of the former elements and functionalities. LCG-2's WMS architecture, which is represented in Figure 2, is highly centralized and each functionality is provided by almost a different machine, as it is conceived as a network service. The EGEE WMS components are the following: The User Interface (UI), which is from where the user sends the jobs; the Resource Broker (RB), which is based in Condor-G and uses most of its functionality; the Computing Element (CE) and the Worker

[4] http://www.viola-testbed.de/

Nodes (WN), which are the cluster frontend and the cluster nodes respectively, as it is established in the fixed configuration dictated by LCG-2's architecture; the Storage Element (SE), which is used for job files storage; the Logging and Bookkeeping service (LB), which registers job events [18].

After the user initiates his use of the Grid by authenticating from the UI and the job parameters are defined, the RB receives the input files through an *input sandbox*. The RB contacts the Information Service, where resource (CE and SE) information is published. If needed by the job, the Catalog Service is also queried by the RB. Then, the Job Submission Service receives an expanded version of the job description, result of the scheduling decisions taken by the RB (matchmaking process), so it submits the job to the chosen CE. In the meanwhile, the RB submits to that CE the *input sandbox* and its contact information. The job then is sent to a WN where it is executed. When the job is done, the *ouput sandbox* is sent back to the RB and then, to the user when requested. In any case, there are several alternative brokering services to submit the job.

Grid*W*ay works on top of Globus services, performing job execution management and resource brokering, allowing unattended, reliable, and efficient execution of jobs, array jobs, or complex jobs on heterogeneous, dynamic and *loosely-coupled* Grids formed by Globus resources. Grid*W*ay's modular architecture, represented in Figure 3, is conformed by the Grid*W*ay Daemon (GWD) and different Middleware Access Drivers (MADs) to access different Grid Services (resource information, job execution and file transfer), all of them in just one host, as Grid*W*ay is conceived as a client tool. Grid*W*ay performs all the job scheduling and submission steps transparently to the end user adopting job execution to changing Grid conditions by providing fault recovery mechanisms, dynamic scheduling, migration on-request and opportunistic migration. A typical job cycle in Grid*W*ay is as explained ahead. After the user provides the GWD with the job description, the resource selection is performed. Once a decision is made, the needed execution and transfer MADs are loaded. The first stage is the *prolog*, where the remote system is prepared by creating a job directory and staging the input files. Then comes the *wrapper* where the real job execution takes place. Finally, during the *epilog* stage, the output files are staged back and the remote directory is cleared. Grid*W*ay allows the deployment of organization-level metaschedulers that provide support for multiple intra-organization users in each scheduling instance. There is one scheduling instance for each organization and all instances compete with each other for the available resources.

4.1 Scheduling Capabilities

Both LCG-2 WMS and Grid*W*ay treat jobs in a FIFO way and support dynamic scheduling, providing a way to filter and evaluate resources based on dynamic attributes, by means of requirement and rank expressions. In LCG-2 the names of these attributes are the same as retrieved from the information service. Nevertheless, in Grid*W*ay, these expressions are based on common resource attributes, independent from the information service, providing another

way of decoupling. While LCG-2 RB accesses only the BDII servers and only processes the GLUE Schema, Grid*Way*'s different information MADs allow to access the most common information services. Considering execution and transfer functionalities, and using the adoption of interfaces and protocols based on Web Services (WS) for this comparison, both LCG-2 and Grid*Way* support Globus pre-WS services, but only Grid*Way* allows access to Globus WS services [19].

Grid*Way* supports opportunistic migration. That is, each scheduling cycle evaluates the benefits of migrating submitted jobs to new available resources (recently added or freed) by comparing rank values. In LCG-2, this functionality is not supported and the ranking only affects submission.

Considering performance slowdown detection, Grid*Way* takes count of the suspension time in remote batch systems and requests a migration when it exceeds a given threshold. Moreover, jobs are submitted together with a lightweight self monitoring system. The job will migrate when it doesn't receive as much CPU as the user expected. None of the performance slowdown detection mechanisms given by Grid*Way* are implemented in LCG-2. The monitoring in the LCG-2 architecture is provided by the LB, which records only basic job states and mixes them with events originated in other components.

With Grid*Way*, an application can take decisions about resource selection as its execution evolves by modifying its requirement and rank expressions and requesting a migration. In LCG-2 RB instead, these expressions are only set at the beginning.

In LCG-2, the PBS Event Logging (APEL) is employed for distributed accounting [20]. Grid*Way* gives the user local accounting functionalities, standing on the Berkeley Database.

4.2 Fault Detection & Recovery Capabilities

The LCG-2 RB incorporates error detection mechanisms provided by Condor-G [21]. On the other hand, Grid*Way* detects job cancellation (when the job exit code is not specified), remote system crash and network disconnection (both when the polling of the job fails). In all of these cases, Grid*Way* requests a migration for the job [22].

LCG-2 WMS supports checkpointing by providing an API to allow applications to be instrumented to save the state of the process (represented by a list of variable and value pairs) at any moment during the execution of a job. Also, it provides the mechanisms to restart the computation from checkpointed data (previously saved state). If a job fails, the WMS automatically reschedules the job and resubmits it to another compatible resource. There, the last state is automatically retrieved and the computation is restarted. The user can also retrieve the saved state for a later manual resubmission, where the user can specify if the job must start from this retrieved checkpoint data. With Grid*Way*, user-level checkpointing or architecture independent restart files managed by the programmer must be implemented. Migration is implemented by restarting the job on the new candidate host. If the checkpointing files are not provided, the

job should be restarted from the beginning. These checkpoints are periodically retrieved to the client machine or a checkpoint server.

Also the system running the scheduler could fail. Grid*Way* persistently saves its state in order to recover or restart the jobs when the system is restarted. LCG-2 RB relies on Condor-G, which stores persistently all relevant state for each submitted job in the scheduler's job queue [10].

4.3 User Interface Functionality

Both Grid*Way* and LCG-2 RB allow single jobs. The LCG-2 RB can handle jobs with dependencies (DAGMan functionality) and interactive jobs. On the other hand, Grid*Way* allows array jobs and jobs with dependencies. For providing this last job type support, Grid*Way* gives the user a functionality to synchronize jobs. In the case of LCG-2, synchronization must be implemented by a periodical polling (active wait), and job dependencies is supported by the Condor's *DAGMan* tool [20]. A *DAGMan* process is locally spawned for each Direct Acyclic Graph (DAG) submitted to Condor-G.

Focusing on the command line interface, both Grid*Way* and LCG-2 RB give the user full control of his jobs. Anyway, Grid*Way* incorporates commands which allow the user to migrate and synchronize jobs, functionalities not provided by LCG-2.

Also, Grid*Way* offers C and Java implementations of the DRMAA Application Programming Interface, which is a Open Grid Forum (OGF) standard [23, 24]. The EDG WMS API given by LCG-2[5] is not standard.

4.4 Installation & Configuration Issues

Both Grid*Way* and LCG-2 RB are modular. The LCG-2 RB is a network service and stands over a great list of external dependencies (requires more other services than Globus): Condor-G, MySQL, etc. Grid*Way* stands on the Globus middleware at the client side and could be extended or used as a building block for more complex architectures that implement service-level agreements (SLAs) or advanced reservation. Also, LCG-2 is encouraged to be installed in Scientific Linux machines. Nevertheless, other flavors of Linux are being used.

Analyzing security mechanisms, LCG-2 takes them from the Globus Toolkit GSI and also implements the VOMS, where an user name and role are used (it manages the authorization information in the scope of the VO). VOMS allows a fine grained control of the use of the resources both to the users' organization and to the resource owners [25].

Grid*Way* can be installed to implement several Grid architectures, namely: enterprise Grids, to enable diverse resource sharing to improve internal collaboration and achieve a better return from IT investment; partner Grids (like the EGEE infrastructure described here) to provide large-scale, secure and reliable sharing of resources among partner organizations; and utility Grids [26] to

[5] http://www.to.infn.it/grid/workload_management/apiDoc/edg-wms-api-index.html

provide access to the latest computing platform and technology and still be flexible enough to adjust capacity as required without needing to purchase costly hardware.

5 Experimental Conditions and Results

The Grid infrastructure used for the experiments is the corresponding to the test Virtual Organization at the Southwest Federation (SWETEST VO) of the EGEE project (Table 1). All Spanish sites are connected by RedIRIS, the Spanish Research and Academic Network, whose interconnection links of the different nodes are shown in Figure 4.

The target application, called *Truba*, performs the tracing of a single ray of a microwave beam launched inside a fusion reactor [27]. Each experiment involves the execution of 50 instances of the *Truba* application. The experiments were performed with a development version of *Truba*, whose average execution time on a Pentium 4o f 3.20 GHz is 9 minutes. *Truba*'s executable file size is 1.8 MB, input file size is 70 KB, and output file size is about 549 KB.

For the EGEE WMS experiments, it has been developed a framework using the lcg2.1.69 User Interface C++ API, which provides support to submit, monitor and control each single ray tracing application to the grid. This framework works in the following way: First of all, a launcher script generates the JDL files needed. Then, the framework launches all the single ray tracing jobs simultaneously, periodically querying each job's state. And finally, it retrieves the job's output. The scheduling decisions are of course delegated to the EGEE WMS.

Grid*W*ay only relies on Globus services, so it could be used in any Grid infrastructure based on the Globus Toolkit, both pre-WS and WS [28]. In the case of EGEE (LCG-2), Globus behaviour has been slightly modified, but it does not loose its main protocols and interfaces, so Grid*W*ay can be used in a standard way to access LCG-2 resources [29].

In both cases, the jobs were submitted from Universidad Complutense de Madrid. The RB employed for the experiments with LCG-2 was located at the IFIC site and used an eager scheduling policy.

5.1 Experimental Results

Table 2 shows a summary of the performance exhibited by the two scheduling systems in the execution of the fusion application. Additionally, Figure 5 shows the mean execution time per site and Figure 6, the mean transfer time per site. As can be seen, Grid*W*ay presents a higher transfer time, because additional jobs are submitted for the *prolog* and *epilog* stages [3]. However, the experiments were conducted with a GridWay version previous to 4.7, in which this issue was solved. On the other hand, the standard deviation of raw performance metrics can be interpreted as an indicator of the heterogeneity in the grid resources and interconnection links [29]. Finally, the lower overhead induced by Grid*W*ay

shows the benefits of its lighter approach and the functionality for performance slowdown detection.

The EGEE WMS spent 195 minutes (3.25 hours) to execute the 50 jobs, giving a productivity equal to 15.38 jobs/hour. Grid*Way* spent 120 minutes (2 hours) to execute the same workload, giving a productivity equal to 25 jobs/hour. The conclusion is that Grid*Way* takes better advantage of the available resources due to its superior scheduling capabilities on dynamic resources. In fact, during the experiments with the EGEE WMS, several problems described before were evidenced. The LCG-2 RB does not provide support for opportunistic migration and slowdown detection, and jobs are assigned to busy resources.

Additionally, the achieved level of parallelism [30] can be obtained by using the following expression:

$$U = \frac{T_{exe}}{T}, \tag{1}$$

being T_{exe} the sum of job execution times and T the turnaround time. The level of parallelism achieved by Grid*Way* was higher than the level achieved by the EGEE WMS (14.91 and 6.89 respectively).

Not all jobs ended successfully at the first try. In the case of the EGEE WMS, 31 jobs were affected and they had to be resubmitted. However, with Grid*Way*, only 1 job failed, but there were 21 migrations mostly due to suspension timeouts (too much delay in a queue), and better resource discovery (too much time allocated to a resource when better resources are waiting to be used).

A methodology to analyze the performance of computational Grids in the execution of high throughput computing applications has been proposed in [31]. This performance model enables the comparison of different platforms in terms of the following parameters: asymptotic performance (r_∞), which is the maximum rate of performance in tasks executed per second, and half-performance length ($n_{1/2}$), which is the number of tasks required to obtain half of the asymptotic performance. A first order characterization of a grid by means of these parameters is:

$$n(t) = r_\infty t - n_{1/2}. \tag{2}$$

Then, the performance of the system, jobs completed per second, can be defined with a finite number of tasks with:

$$r(n) = n(t)/t = \frac{r_\infty}{1 + n_{1/2}/n}, \tag{3}$$

where n is the number of jobs. The parameters of the model, r_∞ and $n_{1/2}$, are obtained by linear fitting to the experimental results obtained in the execution of the applications.

Figure 7 and Figure 8 show the experimental performance obtained with the two workload management systems, along with that predicted by Eq. (2) and Eq. (3). With EGEE WMS, r_∞ was 0.0051 jobs/second (18.19 jobs/hour) and $n_{1/2}$ was 8.33. With Grid*Way*, r_∞ was 0.0079 jobs/second (28.26 jobs/hour) and $n_{1/2}$ was 1.92. From the different values of $n_{1/2}$, it can be deduced that

Grid *W*ay needs less jobs to obtain half of the asymptotic performance due to an earlier job allocation in the resources.

6 Conclusions and Future Work

Metascheduling has been proved to be a needed step in the evolution of grid middleware. In this paper, some implementations of the Grid Metascheduler concept have been reviewed. Due to continuity in the research line, the authors have chosen to compare exhaustively the EGEE WMS and Grid *W*ay. The target application to be ported onto the Grid in order to gain experimental results pertains to fusion physics, a new Grid technology demanding area.

There has been demonstrated that Grid *W*ay offers a Metascheduling alternative for users, application developers and Grid managers, to the LCG-2 RB available in the EGEE distribution. Grid *W*ay provides compatibility to applications with DRM systems that implement the DRMAA standard. The command line interface is similar to that found on local resource managers to submit, kill, migrate, monitor and synchronize single, array and interdependent jobs. Moreover, deployment and maintenance of Grid *W*ay is not only easy and fast, also a wide variety of platforms is supported. Finally, Grid *W*ay achieves lower overhead and higher productivity than the EGEE WMS, mostly because it reduces the number of job submission stages and provides mechanisms, not given by the LCG-2 RB, such as opportunistic migration and performance slowdown detection that considerably improve the usage of the resources. Nevertheless, LCG-2 provides other components that weren't considered in this article, such as data management.

As future work, Grid *W*ay will continue its evolution, standing on its modular architecture. Its commitment is to support the access to the different middleware provided by EGEE (see Figure 9).

7 Acknowledgments

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org/.

The authors would like to thank all the institutions involved in the EGEE project, in particular those who collaborated in the experiments.

References

[1] Yu, J., Buyya, R.: A Taxonomy of Workflow Management Systems for Grid Computing. Journal of Grid Computing **3** (2005) 171–200

[2] Schopf, J.M.: Ten Actions when Superscheduling. Technical Report GFD.4, The Open Grid Forum (2001) Scheduling Working Group.

[3] Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Software – Practice and Experience **34** (2004) 631–651

[4] Baker, M., Buyya, R., Laforenza, D.: Grids and Grid Technologies for Wide-Area Distributed Computing. Software – Practice and Experience **32** (2002) 1437–1466

[5] Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. Intl. J. Supercomputer Applications **11** (1997) 115–128

[6] B. Carpenter, E.: RFC 1958: Architectural Principles of the Internet (1996) Category: Informational.

[7] Rajic, H., Brobst, R., Chan, W., Gardiner, J., Haas, A., Nitzberg, B., Tollefsrud, J.: Distributed Resource Management Application API Specification 1.0. Document GFD.22, The Open Grid Forum (2003) DRMAA Working Group.

[8] Merzky, A., Jha, S., Goodale, T., Shalf, J., Smith, C.: Simple API for Grid Applications – Strawman API. Document RC 4, The Open Grid Forum (2005) Available at http://wiki.cct.lsu.edu/saga/space/start/.

[9] Smith, C.: Open Source Metascheduling for Virtual Organizations with the Community Scheduler Framework. Technical report, Platform Computing Inc. (2003)

[10] 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

[11] Coleman, N., Raman, R., Livny, M., Solomon, M.: Distributed Policy Management and Comprehension with Classified Advertisements. Document UW-CS-TR-1481, University of Wisconsin - Madison Computer Sciences Department (2003)

[12] Thain, D., Tannenbaum, T., Livny, M.: Condor and the Grid. In: Grid Computing. John Wiley & Sons, Inc. (2003) 299–335

[13] Wldrich, O., Ziegler, W., Wieder, P.: A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resources. Technical Report TR-0010, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence (2005)

[14] Erwin, D.W., Snelling, D.F.: UNICORE: A Grid Computing Environment. In: Proc. 7th Intl. Conf. Parallel Processing (Euro-Par 2001). Volume 2150 of Lecture Notes in Computer Science. (2001) 825–834

[15] Venugopal, S., Buyya, R., Winton, L.: A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids. Concurrency and Computation: Practice and Experience **18** (2006) 685–699

[16] de Assunao, M., Nadiminti, K., Venugopal, S., Ma, T., Buyya, R.: An Integration of Global and Enterprise Grid Computing: Gridbus Broker and Xgrid Perspective. In: Proc. 4th International Conference on Grid and Cooperative Computing. Volume 3795 of Lecture Notes in Computer Science. (2005) 406–417

[17] Luther, A., Buyya, R., Ranjan, R., Venugopal, S.: Alchemi: A .NET-Based Enterprise Grid Computing System. In: Proc. 6th International Conference on Internet Computing (ICOMP'05). (2005) 269–278

[18] Campana, S., Litmaath, M., Sciaba, A.: LCG-2 Middleware Overview. Available at https://edms.cern.ch/document/498079/0.1 (2004)

[19] Huedo, E., Montero, R.S., Llorente, I.M.: A Modular Architecture for Interfacing Pre-WS and WS Grid Resource Management. Future Generation Computing Systems **23** (2006) 252–261

[20] Avellino, G., Beco, S., Cantalupo, B., et al: The DataGrid Workload Management System: Challenges and Results. Journal of Grid Computing **2** (2004) 353–367

[21] Morajko, A., Fernandez, E., Fernandez, A., Heymann, E., Senar, M.A.: Workflow Management in the CrossGrid Project. In: Proc. European Grid Conference (EGC2005). Volume 3470 of Lecture Notes in Computer Science. (2005) 424–433

[22] Huedo, E., Montero, R.S., Llorente, I.M.: Evaluating the Reliability of Computational Grids from the End User's Point of View. Journal of Systems Architecture, **52** (2006) 727–736

[23] Herrera, J., Huedo, E., Montero, R.S., Llorente, I.M.: Developing Grid-Aware Applications with DRMAA on Globus-based Grids. In: Proc. 10th Intl. Conf. Parallel Processing (Euro-Par 2004). Volume 3149 of Lecture Notes in Computer Science. (2004) 429–435

[24] Herrera, J., Huedo, E., Montero, R.S., Llorente, I.M.: GridWay DRMAA 1.0 Implementation – Experience Report. Document GFD.104, The Open Grid Forum (2007) DRMAA Working Group.

[25] Alfieri, R., Cecchini, R., et al: From gridmap-file to VOMS: Managing Authorization in a Grid Environment. Future Generation Computing Systems **21** (2005) 549–563

[26] Llorente, I.M., Montero, R.S., Huedo, E., Leal, K.: A Grid Infrastructure for Utility Computing. In: Proc. 3rd International Workshop on Emerging

Technologies for Next-generation GRID (ETNGRID 2006), 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006). IEEE Computer Society (2006) 163–168

[27] Castejon, F., Tereshcenko, M.A., et al.: Electron Bernstein Wave Heating Calculations for TJ-II Plasmas. American Nuclear Society **46** (2004) 327–334

[28] Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Use of Globus Pre-WS and WS Resource Management Services with GridWay. In: Proc. 2nd Workshop on Grid Computing and its Application to Data Analysis (GADA'05) On The Move Federated Conferences. Volume 3762 of Lecture Notes in Computer Science. (2005) 234–243

[29] Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. Journal of Parallel and Distributed Computing **66** (2006) 763–771

[30] Huedo, E., Montero, R.S., Llorente, I.M.: An Evaluation Methodology for Computational Grids. In: Proc. 2005 International Conference on High Performance Computing and Communications (HPCC05). Volume 3726 of Lecture Notes in Computer Science. (2005) 499–504

[31] Montero, R.S., Huedo, E., Llorente, I.M.: Benchmarking of High Throughput Computing Applications on Grids. Parallel Computing **32** (2006) 267–279

Authors Short Biographical Notes

Below, some biographical notes from the authors can be found:

José Luis Vázquez-Poletti received his M.E. in Computer Science (2004) from the Universidad Pontificia de Comillas (UPCo). He is doing his Ph.D. in Computer Architecture at the Universidad Complutense de Madrid (UCM). He is a Teaching Assistant of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM since 2006. His research interests lie mainly in Grid Technology, in particular, metascheduling algorithms and their implementation.

Eduardo Huedo received his M.E. in Computer Science (1999) and Ph.D. in Computer Architecture (2004) from Universidad Complutense de Madrid (UCM). He is an Assistant Professor of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM since 2006. Previously, he was Postdoctoral Researcher in the Advanced Computing Laboratory at Centro de Astrobiología (CSIC-INTA), associated to NASA Astrobiology Institute. His research interests include Performance Management and Tuning, Parallel and Distributed Computing and Grid Technology.

Rubén S. Montero received his B.S. in Physics (1996), M.S. in Computer Science (1998) and Ph.D. in Computer Architecture (2002) from the Universidad Complutense de Madrid (UCM). He is an Associate Professor of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM since 2006. He has held several research appointments at ICASE (NASA Langley Research Center), here he worked on computational fluid dynamics, parallel multigrid algorithms and Cluster computing. Nowadays, his research interests lie mainly in Grid Technology, in particular in adaptive scheduling, adaptive execution and distributed algorithms.

Ignacio M. Llorente received his B.S. in Physics (1990), M.S. in Computer Science (1992) and Ph.D. in Computer Architecture (1995) from the Universidad Complutense de Madrid (UCM). He is Executive M.B.A. by Instituto de Empresa since 2003. He is Full Professor of Computer Architecture and Technology in the Department of Computer Architecture and System Engineering at UCM and Senior Scientist at Centro de Astrobiología (CSIC-INTA), associated to NASA Astrobiology Institute. He has held several appointments since 1997 as a Consultant in High Performance Computing and Applied Mathematics at ICASE (NASA Langley Research Center). His research areas are Information Security, High-Performance Computing and Grid Technology.

Tables

Table 1: EGEE grid resources employed during the experiment.

Site	Processor	Speed (GHz)	Nodes	DRMS
CESGA	Intel Pentium III	1.1	46	PBS
IFAE	Intel Pentium 4	2.8	11	PBS
IFIC	AMD Athlon	1.2	127	PBS
INTA-CAB	Intel Pentium 4	2.8	4	PBS
LIP	Intel Xeon	2.8	25	PBS
PIC	Intel Pentium 4	2.8	172	PBS
USC	Intel Pentium III	1.1	100	PBS

Table 2: Performance metrics for both platforms.

	T_{exe} (m)		T_{xfr} (m)		T	Prod.	Ovh.
	Mean	Dev.	Mean	Dev.	(m)	(j/h)	(m/j)
LCG-2	30.33	11.38	0.42	0.06	195	15.38	1.82
Grid$W$ay	36.80	16.23	0.87	0.51	120	25.00	0.52

Figures

Figures referenced in this paper:

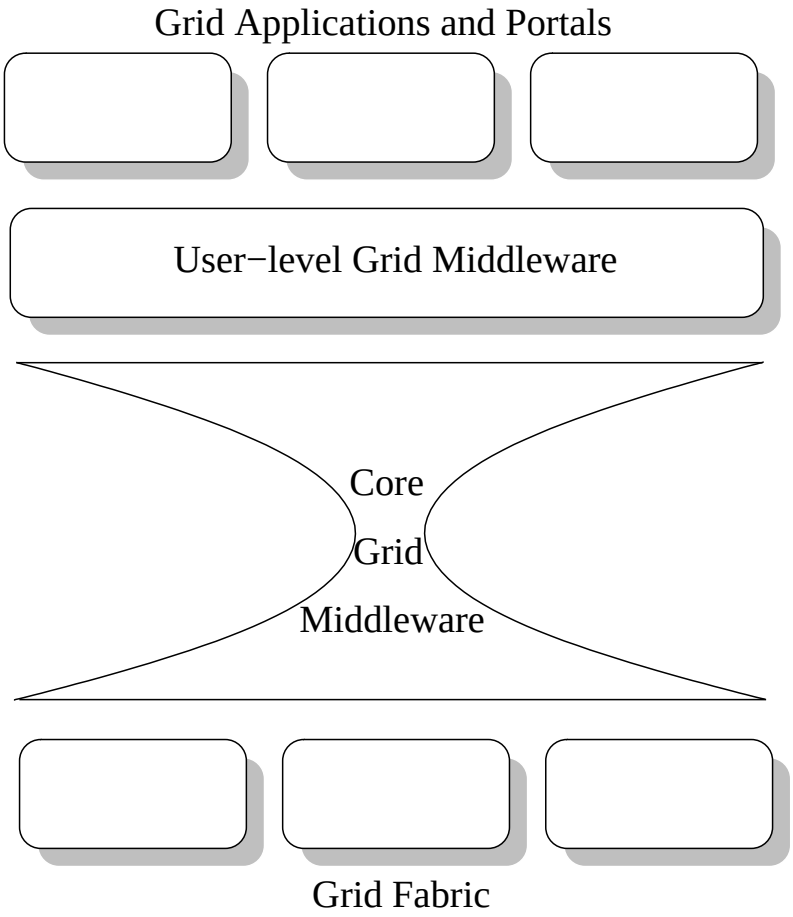

Figure 1: *Loosely-coupled* Grid layers.

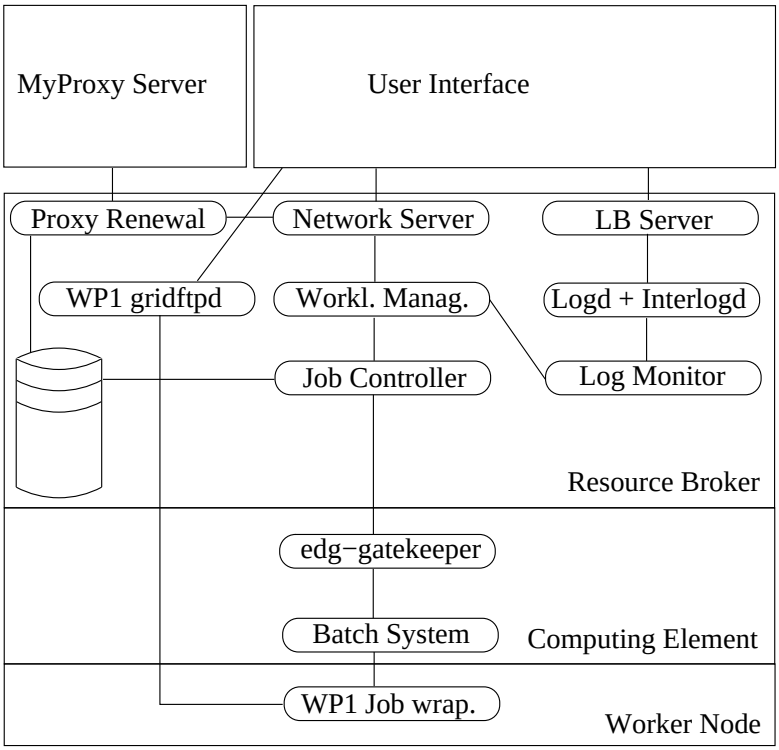

Figure 2: Architecture of LCG-2's WMS.

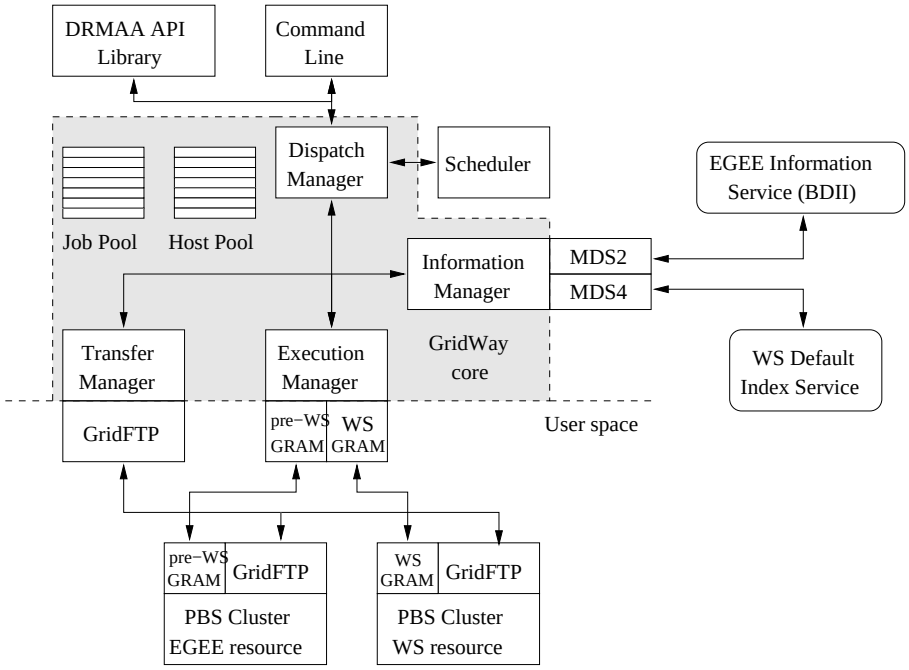

Figure 3: Architecture of Grid *Way*.

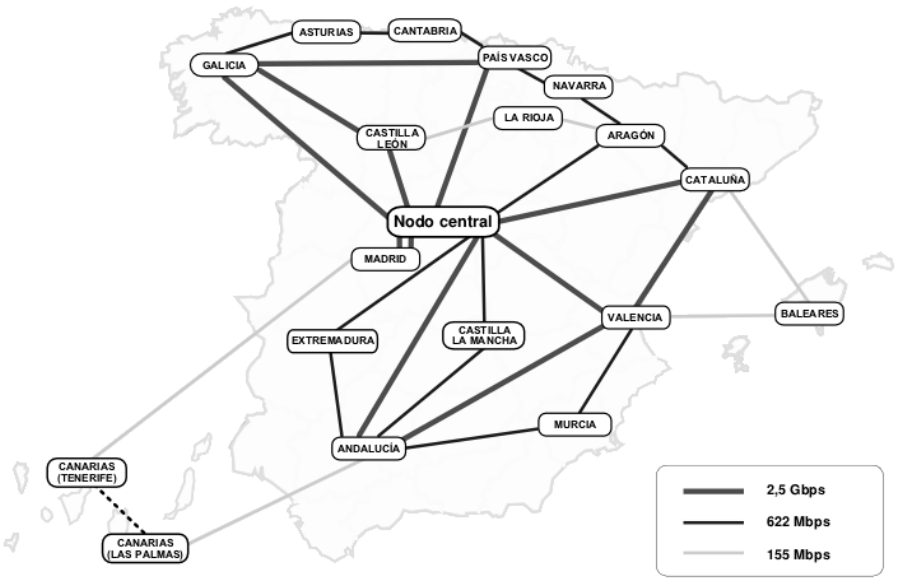

Figure 4: Topology and bandwidths of RedIRIS-2.

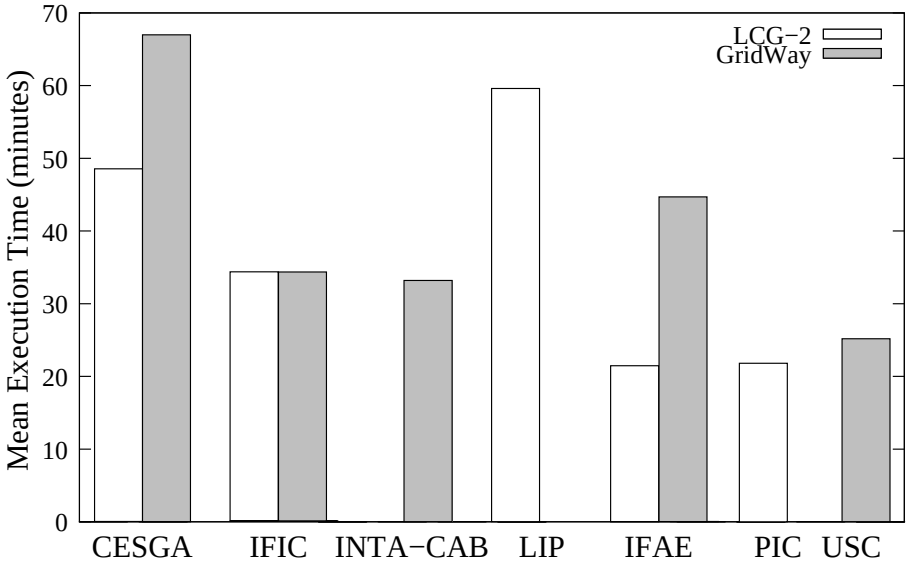

Figure 5: Mean T_{exe} per Site.

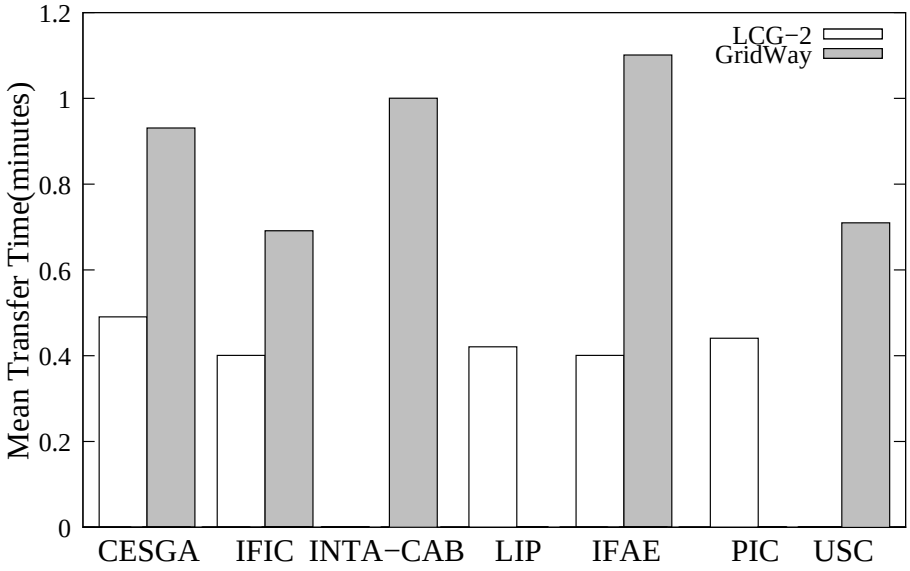

Figure 6: Mean T_{xfr} per Site.

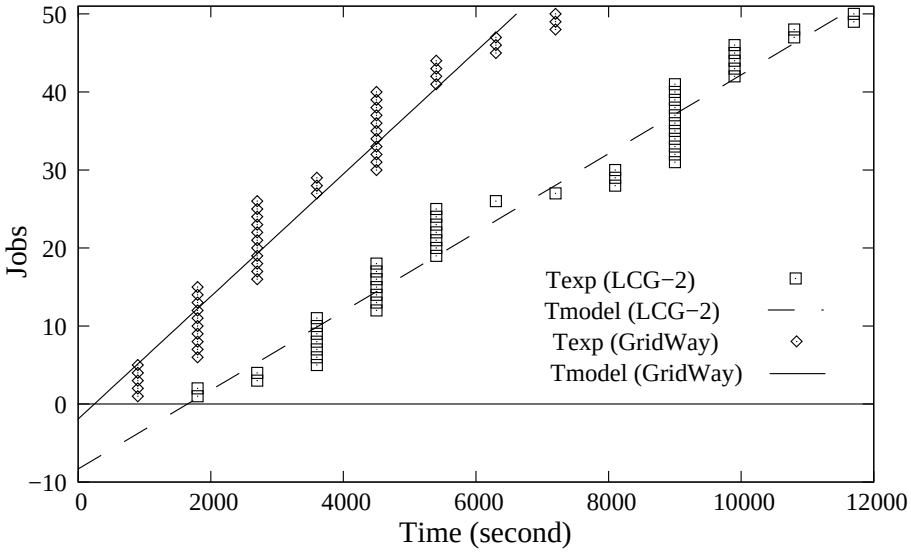

Figure 7: Measurements of r_∞ and $n_{1/2}$ parameters for both platforms.

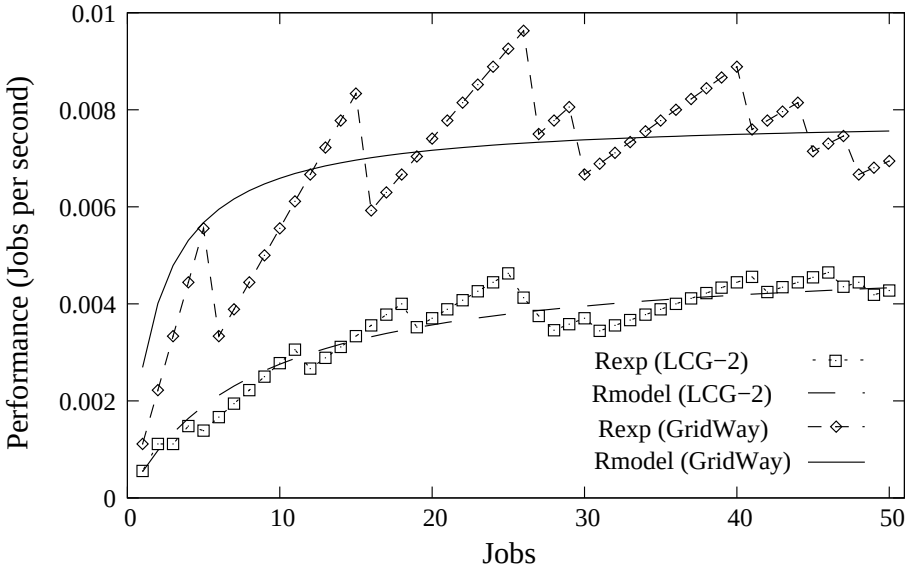

Figure 8: Experimental and predicted performance.

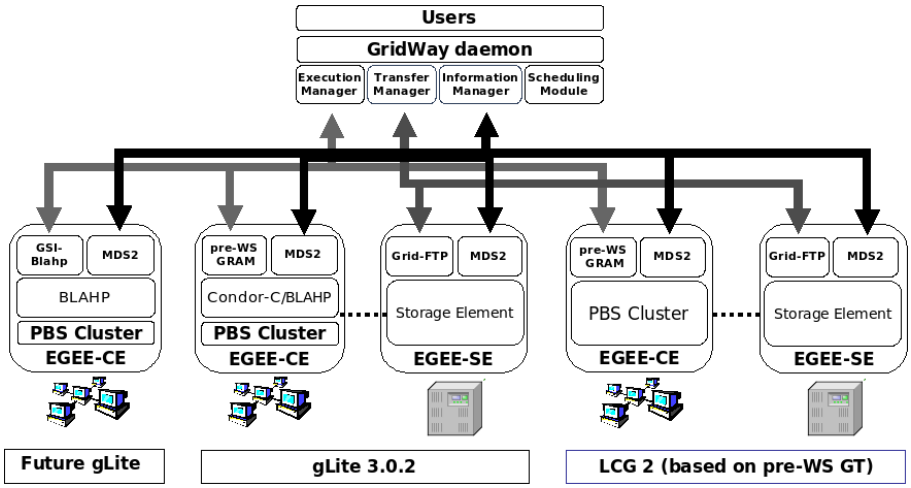

Figure 9: Interaction of Grid*Way* with different EGEE middleware.