A Comparative Analysis between EGEE and Grid *W*ay Workload Management Systems*

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

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

Abstract. Metascheduling is a key functionality of the grid middleware in order to achieve a reasonable degree of performance and reliability, given the changing conditions of the computing environment. In this contribution a comparative analysis between two major grid scheduling philosophies is shown: a semi-centralized approach, represented by the EGEE Workload Management System, and a fully distributed approach, represented by the Grid *W*ay Metascheduler. This comparative is both theoretical, through a functionality checklist, and experimental, through the execution of a fusion plasma application on the EGEE infrastructure.

1 Introduction

The growing computational needs of nowadays projects have permitted the evolution to a new paradigm called Grid Computing. Among the intervening elements of a computional grid, the Metascheduler is gathering most attention as a way to meet these challenging needs. 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].

Several implementations of the Metascheduler can be found, such as CSF, Silver, Nimrod/G, Condor-G, GHS, GrADS, MARS, AppLeS and Gridbus. From these, we would like to remark the following: CSF supports advance reservation booking and offers round-robin and reservation based scheduling algorithms. Scheduling characteristics provided by Nimrod/G strive for the equilibrium between resource providers and resources consumers via auctioning mechanisms [2]. Condor-G is not conceived for supporting scheduling policies but in the other hand, it supplies mechanisms, such as *ClassAd* and *DAGMan*, that may be useful for a metascheduler standing above [3]. GrADS and AppLeS support scheduling mechanisms that take into consideration both application and system level environments [4, 5]. These solutions provide complementary functionality, which

* 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 TIC2003-01321. The authors participate in the EGEE project, funded by the European Union.

focuses on specific application domains and grid middlewares, contributing significant improvement in the field.

The aim of the Grid*W*ay metascheduler [6] is to provide Globus user community with a grid scheduling functionality similar to that found in local DRM (Distributed Resource Management) systems. In a precedent contribution, we showed how a joint testbed of EGEE (LCG-2 middleware) and non-EGEE resources (Globus middleware) can be harnessed by Grid*W*ay with good results [7]. The evaluation of the resulting infrastructure showed that reasonable levels of performance and reliability were achieved. The objective of this contribution is to compare two metascheduling philosophies by means of two representative implementations: EGEE WMS (Workload Management System) and Grid*W*ay Metascheduler. In both cases, the experimental results are obtained on EGEE computing resources.

The structure of this paper is as follows. In Section 2, a short architecture overview of the EGEE WMS and the Grid*W*ay Metascheduler can be found, along with a comparative analysis of the functionality provided by both solutions is shown. Then, Section 3 evaluates the performance obtained by the two alternatives in the execution of fusion plasma application on the EGEE infrastructure. Finally in Section 4, the paper ends up with some conclusions.

2 Grid Scheduling Infrastructures

The EGEE WMS and Grid*W*ay are representative metascheduling technologies of different strategies to deploy a grid scheduling infrastructure. Grid*W*ay follows the *loosely-coupled* Grid principle [8], mainly characterized by: autonomy of the multiple administration domains, heterogeneity, scalability and dynamism. A Grid*W*ay 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 architecture 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 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 [9].

For this contribution's purpose, we have studied the RB, where the user submits the job and its matching is performed. Here, the RB can adopt an eager or lazy policy for scheduling the jobs. While with the eager policy the job will likely end up in a queue, with lazy scheduling the job is held until a resource becomes available. Then the job is sent to the chosen CE and, from there, executed in a corresponding WN. 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 is conformed by the Grid*W*ay Daemon (GWD) and different Middleware Access Drivers (MADs) to access different Grid Services (information, execution and 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 (using a lazy approach) 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. Grid*W*ay allows the deployment of organization-level meta-schedulers 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.

2.1 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*W*ay's different information MADs allow to access the most common information services. Considering execution and transfer functionalities, both LCG-2 and Grid*W*ay support Globus Pre-WS services, but only Grid*W*ay allows access to Globus WS services [10].

Grid*W*ay 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*W*ay 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 light-weight 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*W*ay 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*W*ay, 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.

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*W*ay, 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.

In LCG-2, the PBS Event Logging (APEL) is employed for distributed accounting [11]. Grid*W*ay gives the user local accounting functionalities, standing on the Berkeley Database.

Dependency in job submission is supported in LCG-2 RB by the Condor's DAGMan tool [11]. A DAGMan process is locally spawned for each Direct Acyclic Graph (DAG) submitted to Condor-G. Grid*W*ay also provides support for job dependencies.

2.2 Fault Detection & Recovery Capabilities

The LCG-2 RB incorporates error detection mechanisms provided by Condor-G [12]. On the other hand, Grid*W*ay 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*W*ay requests a migration for the job [13].

Also the system running the scheduler could fail. Grid*W*ay 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 [3].

2.3 User Interface Functionality

Both Grid*W*ay 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*W*ay allows array jobs, jobs with dependencies and complex jobs.

For providing complex job support, Grid*W*ay gives the user a functionality to synchronize jobs. In the case of LCG-2, this must be implemented by a periodical polling (active wait).

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

Also, Grid*W*ay offers C and Java implementations of the DRMAA Application Programming Interface, which is a Global Grid Forum (GGF) standard [14]. The EDG WMS API given by LCG-2[1] is not standard.

3 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 1.

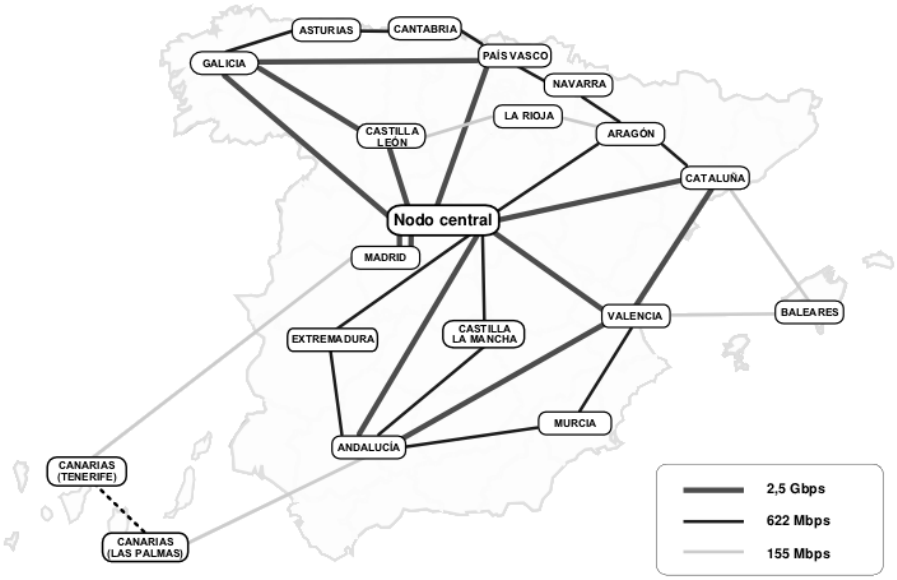

Fig. 1. Topology and bandwidths of RedIRIS-2.

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

The target application, called *Truba*, performs the tracing of a single ray of a microwave beam launched inside a fusion reactor [15]. 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 4 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, we have 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*Way* only relies on Globus services, so it could be used in any Grid infrastructure based on the Globus Toolkit, both PreWS and WS [10]. 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*Way* can be used in a standard way to access LCG-2 resources [7].

Table 1. EGEE grid resources employed during the experiment.

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

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.

3.1 Experimental Results

Table 2 shows a summary of the performance exhibited by the two scheduling systems in the execution of the fusion application. As can be seen, Grid*Way* presents a higher transfer time, because of the reverse-server transferring model used for file staging [6] (which has been replaced in version 4.7 for solving this issue). Moreover, the standard deviation of raw performance metrics can be interpreted as an indicator of the heterogeneity in the grid resources and interconnection links [7]. Finally, the lower overhead induced by Grid*Way* shows the

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

Table 2. Performance metrics for both platforms, times are in minutes and productivity is in jobs/hour

	Execution/Job		Transfer/Job		Turnaround	Productivity	Overhead/Job
Framework	Mean	Dev.	Mean	Dev.			
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

The EGEE WMS spent 195 minutes (3.25 hours) to execute the 50 jobs, giving a productivity equal to 15.38 jobs/hour. Grid*W*ay spent 120 minutes (2 hours) to execute the same workload, giving a productivity equal to 25 jobs/hour. We can conclude that Grid*W*ay 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 [16] 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*W*ay 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*W*ay, 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 [17]. 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, we can define the performance of the system, jobs completed per second, 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 2 and Figure 3 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$W$ay, r_∞ was 0.0079 jobs/second (28.26 jobs/hour) and $n_{1/2}$ was 1.92. From the different values of $n_{1/2}$, we can deduce that Grid$W$ay needs less jobs to obtain half of the asymptotic performance due to an earlier job allocation in the resources.

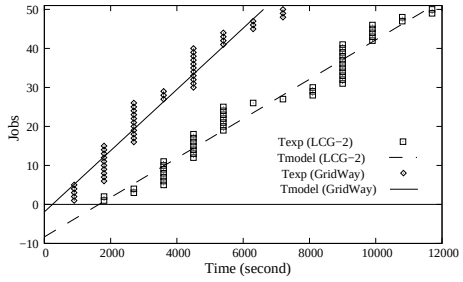

Fig. 2. Measurements of r_∞ and $n_{1/2}$ parameters for both platforms.

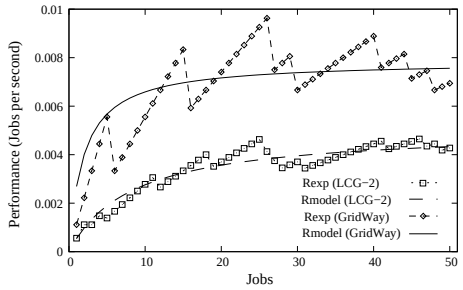

Fig. 3. Experimental and predicted performance.

4 Conclusions

We have demonstrated that Grid$W$ay achieves lower overhead and higher productivity than the EGEE WMS. Grid$W$ay 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 improves

the usage of the resources. Nevertheless, LCG-2 provides other components that weren't considered in this article, such as data management.

5 Acknowledgments

We 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. Buyya, R., Abramson, D., Giddy, J.: Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. Fourth International Conference on High-Performance Computing in the Asia-Pacific Region **1** (2000) 283
3. Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. In: HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01), IEEE Computer Society (2001) 55
4. Dail, H., Sievert, O., Berman, F., Casanova, H., YarKhan, A., Vadhiyar, S., Dongarra, J., Liu, C., Yang, L., Angulo, D., Foster, I.: Scheduling in the Grid Application Development Software Project. In: Grid resource management: state of the art and future trends. Kluwer Academic Publishers (2004) 73–98
5. Berman, F., Wolski, R., Casanova, H., Cirne, W., Dail, H., Faerman, M., Figueira, S., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S., Spring, N., Su, A., Zagorodnov, D.: Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed Systems **14** (2003) 369–382
6. 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
7. Vázquez-Poletti, J., 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
8. Llorente, I.M., Montero, R.S., Huedo, E.: A Loosely Coupled Vision for Computational Grids. IEEE Distributed Systems Online **6** (2005)
9. Campana, S., Litmaath, M., Sciaba, A.: LCG-2 Middleware Overview. Available at https://edms.cern.ch/document/498079/0.1 (2004)
10. 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
11. Avellino, G., Beco, S., Cantalupo, B., et al: The DataGrid Workload Management System: Challenges and Results. Journal of Grid Computing **2** (2004) 353–367
12. 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

13. 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, (2006) (in press).
14. Herrera, J., Montero, R., Huedo, E., Llorente, I.: DRMAA Implementation within the GridWay Framework. In: Workshop on Grid Application Programming Interfaces, 12th Global Grid Forum (GGF12). (2004)
15. Castejon, F., Tereshcenko, M.A., et al.: Electron Bernstein Wave Heating Calculations for TJ-II Plasmas. American Nuclear Society **46** (2004) 327–334
16. 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. Volume 3726 of Lecture Notes in Computer Science. (2005) 499–504
17. Montero, R.S., Huedo, E., Llorente, I.M.: Benchmarking of High Throughput Computing Applications on Grids. Parallel Computing (2006) (in press).