Coordinated Harnessing of the IRISGrid and EGEE Testbeds with Grid*W*ay [⋆]

J. L. Vázquez-Poletti [a], E. Huedo [b], R. S. Montero [a,∗], I. M. Llorente [a,b]

[a] *Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain.*

[b] *Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas, Centro de Astrobiología (CSIC-INTA), 28850 Torrejón de Ardoz, Spain.*

Abstract

Since the late 1990s, we have witnessed an extraordinary development of Grid technologies. Nowadays, different Grid infrastructures are being deployed within the context of growing national and transnational research projects. However, the coexistence of those different infrastructures opens an interesting debate about the coordinated harnessing of their resources, from the end-user perspective, and the simultaneous sharing of resources, from the resource owner perspective. In this paper we demonstrate the efficient and simultaneous use of different Grid infrastructures through a decentralized and "end-to-end" scheduling and execution system. In particular, we evaluate the coordinated use of the EGEE and IRISGrid testbeds in the execution of a Bioinformatics application. Results show the feasibility of building *loosely-coupled* Grid environments only based on Globus services, while obtaining non trivial levels of quality of service, in terms of performance and reliability. Such approach allows a straightforward resource sharing since the resources are accessed by using *de facto* standard protocols and interfaces.

Key words: Grid Computing, Globus Toolkit, Bioinformatics

[⋆] 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.

[∗] Corresponding author. Tel: +34 91 394 75 38. Fax: +34 91 394 46 87.
Email address: `rubensm@dacya.ucm.es` (R. S. Montero).

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 debatable whether 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, while (iii) providing some level of quality of service (QoS), in terms of security, throughput, response time or the coordinated use of different resource types. There is a tendency to ignore the first two requirements in order to get higher levels of quality of service, mainly performance and reliability, for a given application scope. However, those requirements are even more important because they are the key to the success of *the Grid*.

The computational environments created by following the previous Grid philosophy, which we call *loosely-coupled* Grids, resemble the architecture of the Internet. This architecture is based on 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 [2]:

> *"The basic argument is that, as a first principle, certain required end-to-end functions can only be performed correctly by the end-systems themselves."*

These *loosely-coupled* computational environments present the following main characteristics [3]: 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. Finally, the management of resource heterogeneity implies a higher degree of complexity.

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. 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

[1] http://www.globus.org

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.

The coexistence of several projects, each with its own middleware developments, adaptations or extensions, give rise to the idea of coordinated harnessing of resources, or contributing the same resource to more than one project. One approach could be the development of gateways between different middleware implementations [4]. Another approach, more in line with the Grid philosophy, is the development of client tools that can adapt to different middleware implementations. If we consider that nearly all current projects use Globus as the basic Grid middleware, it could be possible a shift of functionality from resources to brokers or clients. This will allow the resources to be accessed in a standard way, making the task of sharing resources between organizations and projects easier.

In this work we demonstrate the "end-to-end" principle in a Grid infrastructure, and so the feasibility of building *loosely-coupled* Grids environments only based on Globus services and user-level middleware, while obtaining non trivial levels of quality of service. This approximation also enables the coordinated harnessing and integration of existing Grid infrastructures. To this end, we consider a testbed built up from resources inside IRISGrid[2], the Spanish National Grid Initiative, and resources inside EGEE (Enabling Grids for E-sciencE)[3], the European production-level Grid infrastructure. We analyze the coordinated use of this testbed with the execution of a Bioinformatics application, since this is one of the most resource-demanding fields. Due to the distributed nature of the Grid, it is very important to quantify the overheads induced by all its components, and to analyze their influence in the global performance. In this way, it is possible to estimate the performance impact of using different information services, schedulers, resource managers...

The structure of the paper is as follows. In Section 2, we will discuss some aspects about Grid infrastructure and middleware. Section 3 reviews a non-exhaustive list of current middleware and infrastructure projects and their relation with the Grid philosophy. The resulting experimental testbed and the target application are described in Sections 4 and 5, respectively, while Section 6 presents the obtained results and evaluates the overall performance of the infrastructure. Finally, Section 7 ends up with some conclusions.

[2] http://irisgrid.rediris.es
[3] http://www.eu-egee.org

2 Grid Infrastructure and Middleware

A Grid infrastructure is usually decomposed in the following layers [3]:

(1) Grid applications and portals.
(2) User-level Grid middleware.
(3) Core Grid middleware.
(4) 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 remain these layers separated and independent with a limited and well defined set of interfaces and protocols between them.

The Globus toolkit [5] has become a *de facto* standard in Grid Computing as core Grid middleware. 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 [6]. However, the user is responsible for manually performing all the submission steps in order to achieve any functionality.

The application of the "end-to-end" principle to Grid computing requires user-level middleware 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 [7].

At the other end, resource management software is advisable in Grid fabric to provide system administrators with tools to determine the amount of resources they are willing to devote to the Grid, avoiding their saturation by Grid jobs. Such kind of software [8] will aid in the expansion of the Grid, leading resource owners to embrace Grid technologies and share their resources with more confidence, because the performance for local users will always be assured. Moreover, the "end-to-end" principle reduces the firewall configuration to a minimum, which is also a welcome advance for security administrators. The more confident resource owners are, the more nodes they will add to the Grid, overcoming the typical scenario where administrators share only a small fraction of their hosts due to their mistrust of the Grid.

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 resource sharing and security policies. Undoubtedly, an approach that allows administrators to have full control of their resources could help to overcome these socio-political difficulties [9].

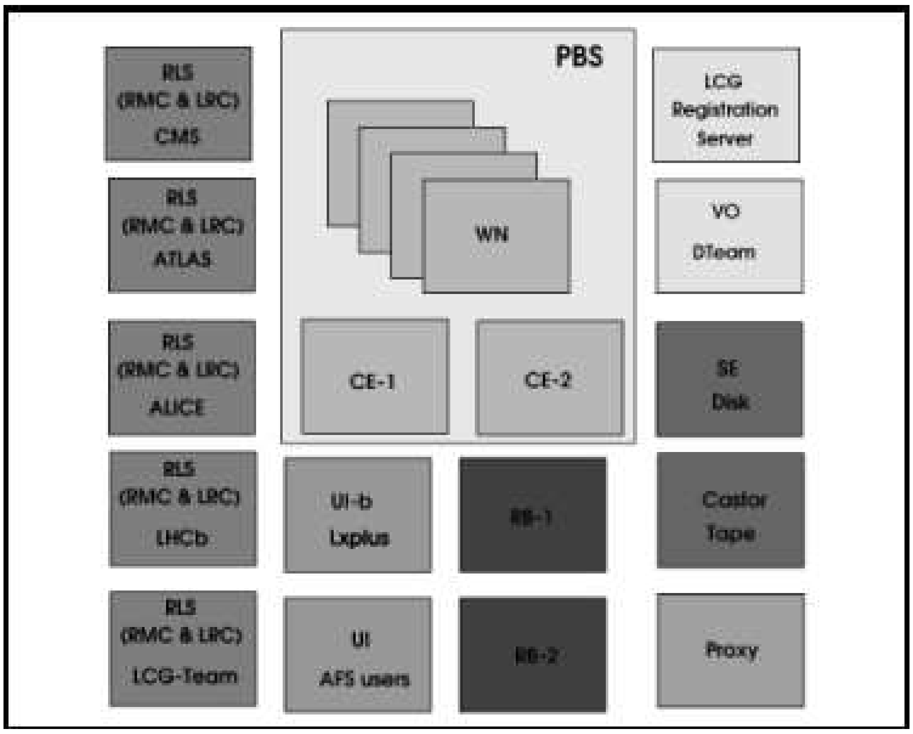

Fig. 1. LCG-2 available services at CERN (each service usually corresponds with a separate server node) [10].

3 The Grid Philosophy in Some Existing Projects

The EGEE (Enabling Grids for E-sciencE) project is creating the larger production-level Grid infrastructure, 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, as being tightly related. EGEE uses the LCG (LHC Computing Grid)[4] middleware, LCG-2 (see Figure 1). This presents some limitations in terms of heterogeneity, as it has a 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. LCG's focus is mainly on particle physics applications that depend on a single organization, CERN (the European Organization for Nuclear Research). Nevertheless, it is expected that the new EGEE middleware, gLite[5], will overcome to some extent, some of these limitations.

On the other hand, the main objective of the IRISGrid initiative is the creation of a stable national Grid infrastructure. IRISGrid‘s first mission is to

[4] http://lcg.web.cern.ch
[5] http://www.glite.org

give the necessary protocols, procedures and guidelines for creating a research Grid within Spain. This initiative seeks to link geographically distant resources so that all the interested groups have access to a research testbed. IRISGrid only defines the core Grid middleware within the Grid fabric, requiring different user-level Grid middleware. The first version of the IRISGrid testbed is based only on Globus, and it has been widely used through the Grid*W*ay framework [6].

In the middle, there are highly interesting projects like Nordugrid [7], whose architectural requirements are very close to that of IRISGrid. Nevertheless, the requirement of being based only on Globus basic services is not considered. NorduGrid defines the user-level and the core Grid middleware, while leaving some flexibility in the Grid fabric [11]. Nevertheless, it presents some interesting benefits like : scalability and no single point of failure; and resources not necessarily dedicated to Grid jobs, but are under the control of their owners and have few site requirements.

In NorduGrid there should be a shared file system (e.g. NFS), the RSL used is an extended one, and there is a Grid Manager (GM) acting as a smart front-end for job submission to a cluster. The GM uses a GridFTP interface for job submission, instead of GRAM, and provides a virtual directory tree, access control based on the user certificate's DN and access to meta-data catalogs. There are some ongoing efforts to provide interoperability between LCG and NorduGrid middlewares.

Finally, Grid*W*ay [12] is a user-level Grid middleware, which uses Globus as core Grid middleware over any Grid fabric. Therefore, Grid*W*ay could be used in any Grid infrastructure that uses Globus, without modifications, as core Grid middleware. In this paper, we will show its utilization in a joint IRISGrid/EGEE testbed, both based on Globus. In the case of EGEE, Globus behaviour is slightly modified, but it doesn't loose its main protocols and interfaces, so it can be used in a standard way. That would be impossible in the case of NorduGrid, since the interface for job management has been radically changed, using a GridFTP interface instead of the standard GRAM interface.

A key aspect in these three alternatives (EGEE, NorduGrid and Grid*W*ay) is how job execution is performed. In EGEE, file transfers are initiated by a job wrapper running in the compute nodes. In NorduGrid, file transfers are initiated by the Grid Manager running in the front-end node, making use of the extended RSL. In Grid*W*ay, job execution is performed in three steps:

(1) *prolog*: It prepares the remote system by creating a experiment directory

[6] http://www.gridway.org
[7] http://www.nordugrid.org

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.

As some other projects [13–17], Grid*W*ay addresses adaptive scheduling, adaptive execution and fault tolerance. However, in the context of this paper, we would like to remark the advantages of its modular, decentralized and "end-to-end" architecture for job adaptation to a dynamic Grid environment.

4 Experimental Testbed

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. . .

Some centers are inside IRISGrid, the Spanish Grid Initiative and Research Testbed, which is composed by around 40 groups from different spanish institutions. In the experiment, 7 of them participated donating a total number of 195 CPUs. The IRISGrid resources are shown in Table 1. Other centers participate in the EGEE project, which is composed by more than 100 contracting and non-contracting partners. In the experiment, 7 spanish centers participated, donating a total number of 333 CPUs. The EGEE resources are shown in Table 2.

Together, the testbed is composed by 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 nodes are shown in Figure 2. Table 3 summarizes the core Grid middleware components used in the experiments.

Next, we describe the modifications performed in both the infrastructure and

Table 1
IRISGrid resources contributed to the experiment.

Name	Site	Network	Processor	Speed	Nodes	RM
heraclito	RedIRIS	Central	Intel Celeron	700MHz	1	Fork
platon	RedIRIS	Central	2×Intel PIII	1.4GHz	1	Fork
descartes	RedIRIS	Central	Intel P4	2.6GHz	1	Fork
socrates	RedIRIS	Central	Intel P4	2.6GHz	1	Fork
aquila	DACYA-UCM	Madrid	Intel PIII	700MHz	1	Fork
cepheus	DACYA-UCM	Madrid	Intel PIII	600MHz	1	Fork
cygnus	DACYA-UCM	Madrid	Intel P4	2.5GHz	1	Fork
hydrus	DACYA-UCM	Madrid	Intel P4	2.5GHz	1	Fork
babieca	LCASAT-CAB	Madrid	Alpha EV67	450MHz	30	PBS
bw	CESGA	Galicia	Intel P4	3.2GHz	80	PBS
llucalcari	IMEDEA	Baleares	AMD Athlon	800MHz	14	PBS
augusto	DIF-UM	Murcia	4×Intel Xeon *	2.4GHz	1	Fork
caligula	DIF-UM	Murcia	4×Intel Xeon *	2.4GHz	1	Fork
claudio	DIF-UM	Murcia	4×Intel Xeon *	2.4GHz	1	Fork
lxsrv1	BIFI-UNIZAR	Aragón	Intel P4	3.2GHz	50	SGE

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

Table 2
EGEE resources contributed to the experiment.

Name	Site	Network	Processor	Speed	Nodes	RM
ce00	LCASAT-CAB	Madrid	Intel P4	2.8GHz	8	PBS
mallarme	CNB	Madrid	2×Intel Xeon	2.0GHz	8	PBS
lcg02	CIEMAT	Madrid	Intel P4	2.8GHz	6	PBS
grid003	FT-UAM	Madrid	Intel P4	2.6GHz	49	PBS
gtbcg12	IFCA	Cantabria	2×Intel PIII	1.3GHz	34	PBS
lcg2ce	IFIC	Valencia	AMD Athlon	1.2GHz	117	PBS
lcgce02	PIC	Cataluña	Intel P4	2.8GHz	69	PBS

Grid*W*ay. 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 resources. Regarding authorization, we had to add an entry for the user in the

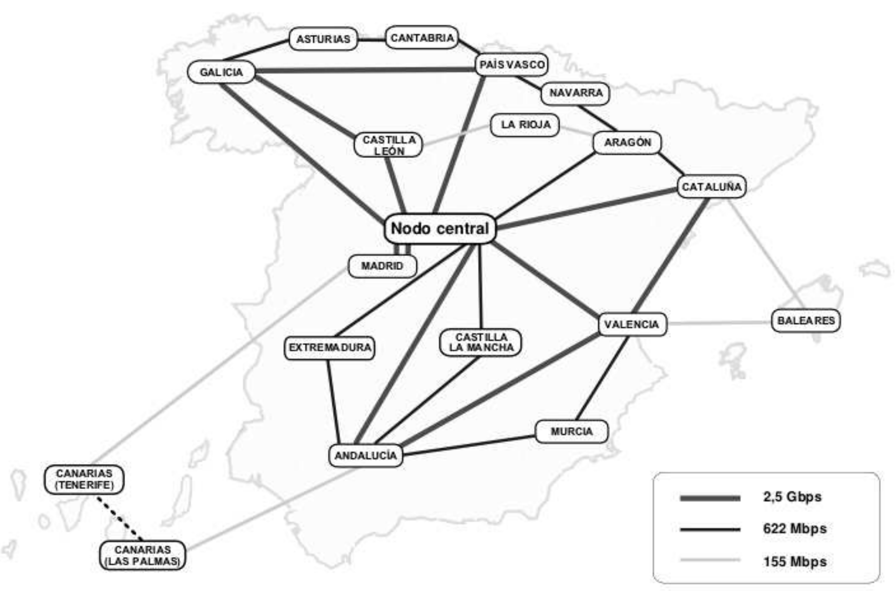

Fig. 2. Topology and bandwidths of RedIRIS-2.

Table 3
Core Grid middleware.

Globus 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

`grid-mapfile` in both IRISGrid and EGEE resources.

Another possibility would be to use two different user certificates, each one to access each 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 [4].

We also had to introduce some changes in some Grid*W*ay modules. For example, our *resource selector* module only understood the Globus MDS schema, so we had to develop another one understanding the GLUE schema used in EGEE. We also developed a *super resource selector* to merge results from the other two. The access to GIIS (Grid Information Index Service) and BDII (Berkeley Database Information Service) servers is homogeneous, since they both use an LDAP interface for access. The main difference between BDII and GIIS is that the information in BDII is stored in a persistent database, which is periodically updated. We have experienced that BDII can be much more reliable and efficient than GIIS.

The *wrapper* module was also modified to perform an explicit file staging between the front-end and the compute nodes in EGEE clusters, so it has to detect if a shared file system between compute and worker nodes exists or not.

It is interesting to remark the advantages of a modular architecture in the design of Grid*W*ay, since we didn't have to directly modify the source code of the *submission agent*. Instead, we had to only modify two modules that were implemented as scripts, so the modifications were straightforward.

5 The Bioinformatics Application

Bioinformatics, which has to do with the management and analysis of huge amounts of biological data, could enormously benefit from the suitability of the Grid to execute high-throughput applications. It is foreseeable that the Grid will be inevitably adopted, given that biological data is growing very fast, due to the proliferation of automated high-throughput experimental techniques and organizations dedicated to Biotechnology. Therefore, the resources required to manage and analyze this data will be soon only available from the Grid [18].

One of the main challenges in Computational Biology concerns the analysis of the huge amount of protein sequences provided by genomic projects at an ever increasing pace. The structure and function of a protein is coded in its amino acid sequence, but deciphering it has turned out to be a very difficult problem, which is still waiting for a complete solution. Nevertheless, in several cases, particularly when homologous proteins are known, computational methods can be quite reliable.

In the following experiment, 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) [8], aligns with gaps the target sequence with all the 6150 non-redundant struc-

[8] http://PredictionCenter.llnl.gov/casp5/

tures in the Protein Data Bank (PDB)[9], 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 [19].

To speed up the analysis and reduce the data needed, the PDB files are preprocessed to extract the contact matrices, which provide a reduced representation of protein structures. The algorithm is then applied twice, the first time as a fast search, in order to select the 200 best candidate structures, and the second time with parameters allowing a more accurate search of the optimal alignment.

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. If a representative structure of this set is known, the algorithm predicts it as the correct structure. The biological results of the comparative study of several families of orthologous proteins are presented elsewhere [20].

The experiment files consist of: the executable (0.5MB) provided for all the resource architectures in the testbed, the PDB files shared and compressed (12.2MB) to reduce the transfer time, the parameter files (1KB), and the file with the sequence to be analyzed (1KB). The final name of the executable and the file with the sequence to be analyzed is obtained at runtime for each task and each host, respectively.

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 case of protein). Five experiments were conducted on different days during the same week, as shown in Table 4. The average turnaround time for the five experiments was 43.37 minutes.

Tables 5 and 6 reflect the transfer and execution times employed by each host to solve one task of the problem. Same is Table 7 but for each site. These metrics are useful to evaluate the impact of data movement strategies (file re-usage, replica selection and dissemination…), individual resource performance or the influence of the interconnection network. Given the dynamic

[9] http://www.pdb.org

Table 4
Experiments performed.

Date (dd/mm/yy)	Start time (h:mm:ss)	End time (h:mm:ss)	Turnaround time (min)	Throughput (jobs/min)
22/10/2004	20:44:45	21:26:45	42.00	1.90
23/10/2004	23:58:32	0:32:22	33.82	2.37
24/10/2004	1:38:32	2:24:35	46.05	1.74
26/10/2004	13:44:03	14:34:25	50.37	1.59
26/10/2004	19:07:16	19:51:50	44.57	1.80

nature of Grid environments it is important to quantify the fluctuations on these measurements. Thus, the standard deviation of the average of the transfer and execution times can be used as an indicator of the dynamism (if calculated on the same resource over time) and heterogeneity (if calculated at the same time over different resources) of the environment. In Tables 5 and 6, the standard deviation is an indicator of the dynamism, since we calculate it in a aggregated way for the whole five experiments. In Table 7, it indicates both the dynamism and the heterogeneity of the resources inside each site.

As can be seen in Tables 5 and 6, resources with fast processors present a lower mean execution time ($\mu_{lxsrv1} = 687$, $\mu_{bw} = 697$, $\mu_{grid003} = 739$ and $\mu_{lcgce02} = 777$). Some resources also present a higher deviation in the execution time due to the queue system overhead and a slow front-end node ($\sigma_{lcg02} = 342$, $\sigma_{ce00} = 267$ and $\sigma_{lcg2ce} = 226$). The use of SMP nodes also causes a higher variability in the execution time when all the CPUs are simultaneously used ($\sigma_{platon} = 275$, $\sigma_{augusto} = 233$, $\sigma_{caligula} = 233$ and $\sigma_{claudio} = 184$) due to the contended use of shared resources, like memory and system buses. In the case of DIF-UM, all its three resources present two physical CPUs, but they report four logical CPUs due to hyper-threading. Therefore, four jobs were simultaneously scheduled to these resources, resulting in a great performance loss ($\mu_{augusto} = 1200$, $\mu_{caligula} = 1242$, $\mu_{claudio} = 1228$).

As can be seen in Table 7, sites with high heterogeneity present a higher deviation in the execution time ($\sigma_{DACyA-UCM} = 630$ and $\sigma_{RedIRIS} = 393$). Regarding transfer times, sites well connected with the client machine at DACyA-UCM present a lower mean ($\mu_{DACyA} = 63$, $\mu_{RedIRIS} = 62$ and $\mu_{FT-UAM} = 90$) unless they have a slow front-end ($\mu_{LCASAT-CAB} = 216$, $\mu_{IFIC} = 208$, and $\mu_{CIEMAT} = 158$). Sites with worse connection to the client present a higher mean transfer time ($\mu_{IFCA} = 261$, $\mu_{IMEDEA} = 169$, $\mu_{BIFI-UNIZAR} = 152$ and $\mu_{DIF-UM} = 142$) and variability ($\sigma_{IFCA} = 105$, $\sigma_{IMEDEA} = 92$, $\sigma_{BIFI-UNIZAR} = 92$, and $\sigma_{DIF-UM} = 108$).

During the whole time employed for each experiment, some of the executions

Table 5
Execution and transfer times (in seconds) per job on each IRISGrid resource.

	Execution Time		Transfer Time	
Host	Mean	Dev.	Mean	Dev.
heraclito	2146	57	151	107
platon	919	275	67	50
descartes	611	33	48	30
socrates	647	103	51	27
aquila	1895	235	143	93
cepheus	2022	112	64	24
cygnus	755	74	33	20
babieca	1798	28	131	131
bw	697	176	123	54
llucalcari	1567	168	169	92
augusto	1200	233	89	61
caligula	1242	233	153	109
claudio	1228	184	187	131
lxsrv1	687	190	152	92

Table 6
Execution and transfer times (in seconds) per job on each EGEE resource.

	Execution Time		Transfer Time	
Host	Mean	Dev.	Mean	Dev.
ce00	929	267	220	80
mallarme	945	191	123	68
lcg02	932	342	158	115
grid003	739	63	90	67
gtbcg12	1002	52	261	105
lcg2ce	889	226	208	113
lcgce02	777	179	98	68

failed or were suspended for a long time. Grid*W*ay migrated these failed or suspended jobs to other resources. Figure 3 shows the distribution of total executions and failures over sites.

It is of crucial interest to analyze the potential gain in performance that a site

Table 7
Execution and transfer times (in seconds) per job on each site.

	Execution Time		Transfer Time	
Site	Mean	Dev.	Mean	Dev.
RedIRIS	830	393	62	47
DACyA-UCM	1271	630	63	61
LCASAT-CAB	970	321	216	83
CESGA	697	176	123	54
IMEDEA	1567	168	169	92
DIF-UM	1223	218	142	108
BIFI-UNIZAR	687	190	152	92
CNB	945	191	123	68
CIEMAT	932	342	158	115
FT-UAM	739	63	90	67
IFCA	1002	52	261	105
IFIC	889	226	208	113
PIC	777	179	98	68

could obtain by joining the Grid. To this end, let us define Grid speedup as:

$$S_{Site} = \frac{T_{Site}}{T_{Grid}}, \tag{1}$$

where T_{Grid} is the turnaround time using all the Grid, and T_{Site} is the turnaround time using only the resources available in a given site. We have estimated the value of T_{Site} being the maximum theoretical turnaround time for the site:

$$T_{Site} = N \cdot \sum_{i \in Site} \frac{\overline{T}_i^{exe}}{CPU_i}, \tag{2}$$

where N is the number of jobs of the experiment, in this case 80, $\overline{T}_i^{exe}$ is the average execution time per job in resource i, as shown in Tables 5 and 6, and CPU_i is the number of CPUs in resource i, as shown in Tables 1 and 2, with a maximum of four CPUs.

Figure 4 shows each site speedup (the bars) and turnaround time (the values on top of bars). Notice that, since we have limited to four the number of jobs simultaneously submitted to the same resource, these metrics don't show the actual performance gain that could be obtained when using all the

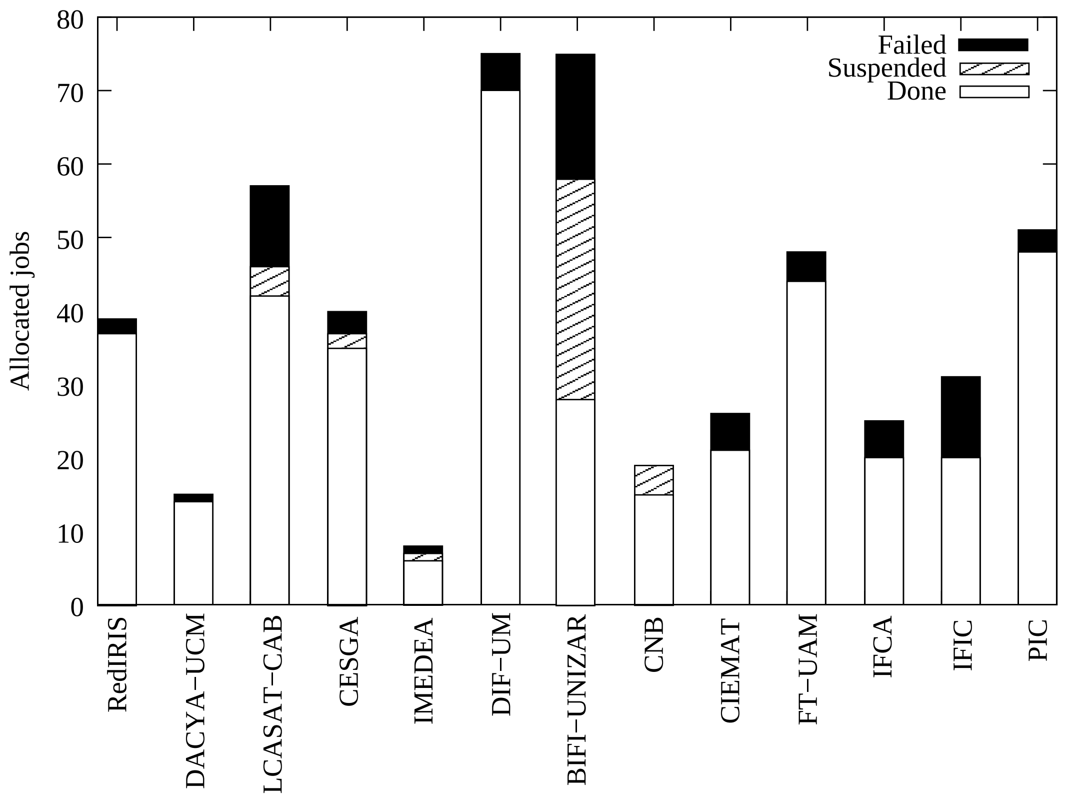

Fig. 3. Aggregated scheduling performed by Grid*W*ay during the five experiments.

resources available on each organization. For example, only four jobs were simultaneously submitted to lcgce02 at PIC, while twelve jobs were simultaneously submitted to augusto, caligula and claudio at DIF-UM (i.e. four jobs to each resource, since they report four CPUs due to hyper-threading).

Figure 5 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 above limitation in the number of simultaneously running jobs in a resource). The throughput achieved on each experiment varies considerably. This is due to the dynamic availability and load of the testbed. For example, resource ce00 at 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.

7 Conclusions

Loosely-coupled Grids allow a straightforward resource sharing since resources are accessed and exploited through de facto standard protocols and interfaces,

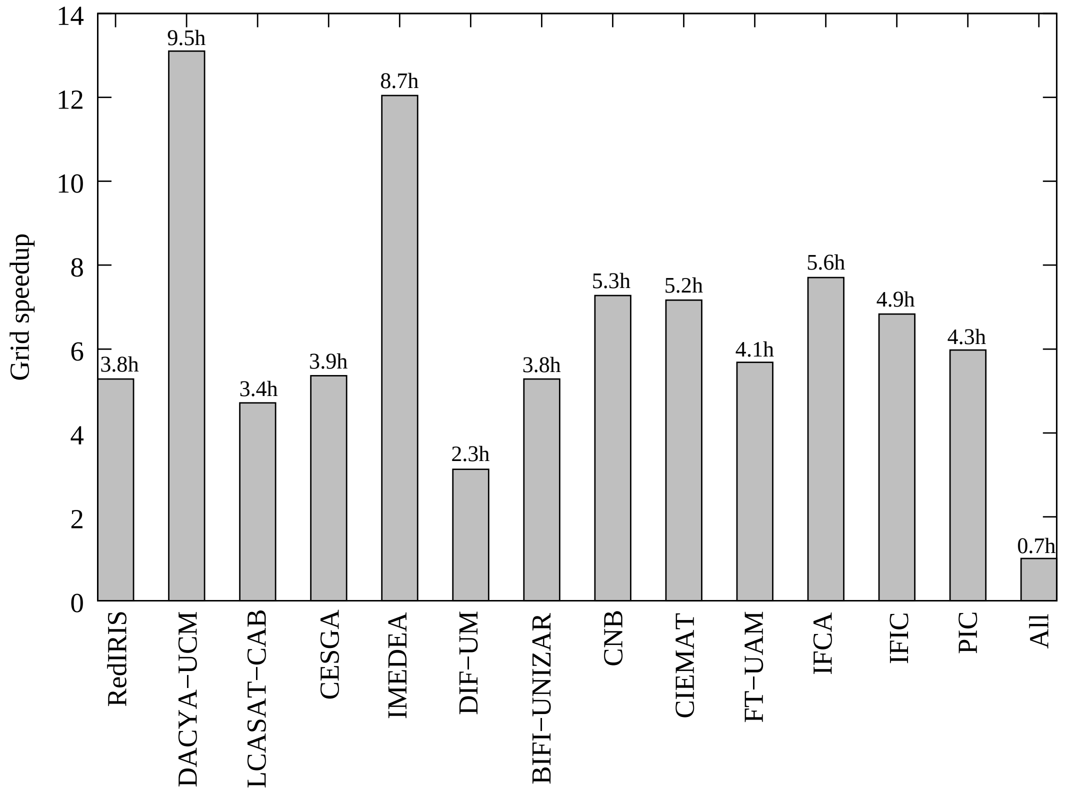

Fig. 4. Grid speedup (S_{Site}) and estimated turnaround time (T_{Site}) for each site.

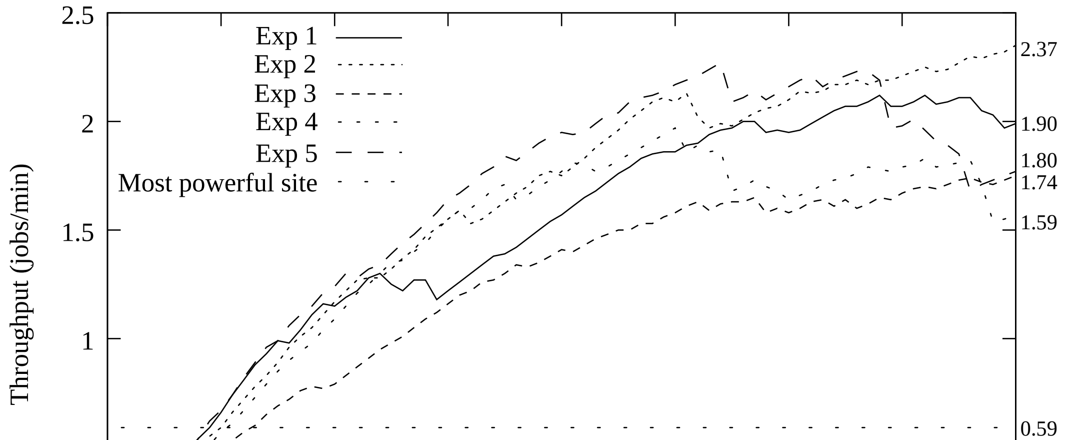

Fig. 5. Testbed dynamic throughput during the five experiments and theoretical throughput of the most powerful site (DIF-UM).

similar to the early stages of the Internet. This way, the *loosely-coupled* model allows an easier, scalable and compatible deployment.

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, is able to work with a standard core Grid middleware over any Grid fabric in a *loosely-coupled* way. Moreover, since similar experiments have been previously performed at the resource side (i.e. the Grid fabric), we can conclude that the "end-to-end" principle could work on the two sides.

The straightforward process of integration of these 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, the evaluation of the resulting infrastructure shows that reasonable levels of performance and reliability are achieved.

Acknowledgements

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. They are Red Académica y de Investigación Nacional (RedIRIS), Departamento de Arquitectura de Computadores y Automática (DACyA) at Universidad Complutense de Madrid (UCM), Laboratorio de Computación Avanzada, Simulación y Aplicaciones Telemáticas (LCASAT) at Centro de Astrobiología (CAB), Centro de Supercomputación de Galicia (CESGA), Instituto Mediterráneo de Estudios Avanzados (IMEDEA), Facultad de Informática (DIF) at Universidad de Murcia (UM), Instituto de Biocomputación y Física de Sistemas Complejos (BIFI) at Universidad de Zaragoza (UNIZAR), Instituto de Física de Cantabria (IFCA), Instituto de Física Corpuscular (IFIC), Centro Nacional de Biotecnología (CNB), Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Departamento de Física Teórica (FT) at Universidad Autónoma de Madrid (UAM) and Port d'Informació Científica (PIC).

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, for his support on understanding and modifying the application.

References

[1] I. Foster, What Is the Grid? A Three Point Checklist, GRIDtoday 1 (6), available at http://www.gridtoday.com/02/0722/100136.html.

[2] E. B. Carpenter, RFC 1958: Architectural Principles of the Internet, category: Informational (Jun. 1996).

[3] M. Baker, R. Buyya, D. Laforenza, Grids and Grid Technologies for Wide-Area Distributed Computing, Software – Practice and Experience 32 (15) (2002) 1437–1466.

[4] R. J. Allan, J. Gordon, A. McNab, S. Newhouse, M. Parker, Building Overlapping Grids, Tech. rep., University of Cambridge (Oct. 2003).

[5] I. Foster, C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, Intl. J. Supercomputer Applications 11 (2) (1997) 115–128.

[6] J. M. Schopf, Ten Actions when Superscheduling, Tech. Rep. GFD-I.4, Scheduling Working Group – The Global Grid Forum (2001).

[7] J. Herrera, E. Huedo, R. S. Montero, I. M. Llorente, Developing Grid-Aware Applications with DRMAA on Globus-Based Grids, in: Proc. 10th Intl. Conf. Parallel and Distributed Processing (Euro-Par 2004), Vol. 3149 of LNCS, 2004, pp. 429–435.

[8] O. S. José, L. M. Suárez, E. Huedo, R. S. Montero, I. M. Llorente, Resource Performance Management on Computational Grids, in: Proc. 2nd Intl. Symp. Parallel and Distributed Computing (ISPDC 2003), 2003, pp. 215–221.

[9] J. M. Schopf, B. Nitzberg, Grids: The Top Ten Questions, Scientific Programming, special issue on Grid Computing 10 (2) (2002) 103–111.

[10] A. Delgado, P. Méndez, F. Donno, A. Sciabà, S. Campana, R. Santinelli, LCG-2 User Guide, Available at http://edms.cern.ch/file/454439/1 (2004).

[11] O. Smirnova, P. Eerola, T. Ekelöf, et al., The NorduGrid Architecture and Middleware for Scientific Applications, in: Proc. 11th Intl. Conf. Computational Science, Vol. 2657 of LNCS, 2003, pp. 264–273.

[12] E. Huedo, R. S. Montero, I. M. Llorente, A Framework for Adaptive Execution on Grids, Intl. J. Software – Practice and Experience (SPE) 34 (7) (2004) 631–651.

[13] F. Berman, R. Wolski, H. Casanova, et al., Adaptive Computing on the Grid Using AppLeS, IEEE Trans. Parallel and Distributed Systems 14 (4) (2003) 369–382.

[14] R. Buyya, D.Abramson, J. Giddy, A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker, Future Generation Computer Systems 18 (8) (2002) 1061–1074.

[15] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor/G: A Computation Management Agent for Multi-Institutional Grids, Cluster Computing 5 (3) (2002) 237–246.

[16] G. Lanfermann, G. Allen, T. Radke, E. Seidel, Nomadic Migration: A New Tool for Dynamic Grid Computing, in: Proc. 10th Symp. High Performance Distributed Computing (HPDC10), IEEE CS, 2001, pp. 429–430.

[17] S. Vadhiyar, J. Dongarra, A Performance Oriented Migration Framework for the Grid, in: Proc. 3rd Intl. Symp. Cluster Computing and the Grid (CCGrid 2003), IEEE CS, 2003, pp. 130–137.

[18] E. Huedo, U. Bastolla, R. S. Montero, I. M. Llorente, Computational Proteomics on the Grid, New Generation Computing 22 (2) (2004) 191–192.

[19] U. Bastolla, M. Vendruscolo, E. W. Knapp, A Statistical Mechanical Method to Optimize Energy Parameters for Protein Folding, Proc. National Academy of Sciences (PNAS) of USA 97 (2000) 3977–3981.

[20] R. van Ham, J. Kamerbeek, C. Palacios, C. Rausell, F. Abascal, U. Bastolla, J. M. Fernandez, L. Jimenez, M. Postigo, F. Silva, J. Tamames, E. Viguera, A. Latorre, A. Valencia, F. Morán, A. Moya, Reductive Genome Evolution in Buchnera Aphidicola, Proc. National Academy of Sciences (PNAS) of USA 100 (2003) 581–586.