Scalable Computing: Practice and Experience
Volume 7, Number 2, pp. 25–32. http://www.scpe.org
ISSN 1895-1767

BENCHMARKING OF A JOINT IRISGRID/EGEE TESTBED WITH A BIOINFORMATICS APPLICATION

J. HERRERA* , E. HUEDO* , R. S. MONTERO* , AND I. M. LLORENTE*

Abstract. Loosely-coupled Grid environments, based only on Globus services, allow a straightforward resource sharing, as the resources are accessed by using *de facto* standard protocols and interfaces, while providing non trivial levels of quality of service. This paper describes the execution of a Bioinformatics application over a highly distributed and heterogeneous testbed. This testbed is composed of resources devoted to EGEE and IRISGrid projects and has been integrated by taking advantage of the modular, decentralized and "end-to-end" architecture of the GridWay framework. Results obtained from the experiments have been analyzed using a performance model for high throughput computing applications.

1. Introduction. Different Grid infrastructures are being deployed within growing national and transnational research projects. The final goal of these projects is to provide the end user with much higher performance than that achievable on any single site. However, from our point of view, it is arguable that some of these projects embrace the Grid philosophy, and to what extent. This philosophy, proposed by Foster [7], defines *a grid* as a system (i) not subject to a centralized control and (ii) based on standard, open and general-purpose interfaces and protocols, (iii) while providing some level of quality of service (QoS), in terms of security, throughput, response time or the coordinated use of different resource types. In current projects, there is a tendency to ignore the first two requirements in order to get higher levels of QoS. However, these requirements are even more important because they are the key to the success of *the Grid*.

The Grid philosophy leads to computational environments, which we call *loosely-coupled* grids, mainly characterized by [3]: autonomy (of the multiple administration domains), heterogeneity, scalability and dynamism. In a loosely-coupled grid, the different layers of the infrastructure should be separated from each other, being only communicated with a limited and well defined set of interfaces and protocols. This layers are [3]: Grid fabric, core Grid middleware, user-level Grid middleware, and Grid applications.

The coexistence of several projects, each with its own middleware developments, adaptations or extensions, arise the idea of using them simultaneously (from an user's viewpoint) or contribute the same resources to more than one project (from an administrator's viewpoint). One approach could be the development of gateways between different middleware implementations [1]. Other approach, more in line with the Grid philosophy, is the development of client tools that can adapt to different middleware implementations. We hope this could lead to a shift of functionality from resources to brokers or clients, allowing the resources to be accessed in a standard way and easing the task of sharing resources between organizations and projects. We should consider that the Grid not only involves the technical challenge of constructing and deploying this vast infrastructure, it also brings up other issues related to security and resource sharing policies [13] as well as other socio-political difficulties [15].

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 [2]. Therefore, instead of succumbing to the temptation of tailoring the core Grid middleware to our needs (since in such case the resulting infrastructure would be application specific), or homogenizing the underlying resources (since in such case the resulting infrastructure would be a highly distributed cluster), we propose to strictly follow the "end-to-end" principle. Clients should have access to a wide range of resources provided through a limited and standardized set of protocols and interfaces. In the Grid, these are provided by the core Grid middleware, Globus, just as, in the Internet, they are provided through the TCP/IP protocols. Moreover, the "end-to-end" principle reduces the firewall configuration to a minimum, which is also welcome by the security administrators.

One of the most ambitious projects to date is EGEE[2] (Enabling Grids for E-sciencE), which is creating a production-level Grid infrastructure providing a level of performance and reliability never achieved before. EGEE currently uses the LCG[3] (LHC Computing Grid) middleware, which is based on Globus. Other much

*Departamento de Arquitectura de Computadores y Automática. Facultad de Informática, Universidad Complutense de Madrid. 28040 Madrid, Spain. {jherrera, ehuedo}@fdi.ucm.es, {rubensm, llorente}@dacya.ucm.es

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

[2] http://www.eu-egee.org

[3] http://lcg.web.cern.ch

TABLE 2.1
IRISGrid and EGEE resources contributed to the experiment.

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

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

more modest project is IRISGrid[4] (the Spanish Grid Initiative), whose main objective is the creation of a stable national Grid infrastructure. The first version of the IRISGrid testbed is based only on Globus services, and it has been widely used through the Grid*W*ay framework[5].

For the purposes of this paper we have used a Globus-based testbed to run a Bioinformatics application through the Grid*W*ay framework. This testbed was built up from resources inside IRISGrid and EGEE projects. The aim of this paper is to demonstrate the application of an "end-to-end" principle in a Grid infrastructure, and the feasibility of building loosely-coupled Grid environments based only on Globus services, while obtaining non trivial levels of quality of service through an appropriate user-level Grid middleware.

The structure of the paper follows the layered structure of Grid systems, from bottom-up. The Grid fabric is described Section 2. Section 3 describes the core Grid middleware. Section 4 introduces the functionality and characteristics of the Grid*W*ay framework, used as user-level Grid middleware. Section 5 describes the target application. Section 6 presents the experimental results, which are analyzed through a benchmarking model in Section 7. Finally, Section 8 ends up with some conclusions.

2. Grid Fabric: IRISGrid and EGEE resources. This work has been possible thanks to the collaboration of those research centers and universities that temporarily shared some of their resources in order to set up a geographically distributed testbed. The testbed results in a very heterogeneous infrastructure, since it presents several middlewares, architectures, processor speeds, resource managers (RM), network links, etc. A brief description of the participating resources is shown in Table 2.1.

Some centers are inside IRISGrid, which is composed of around 40 research groups from different spanish institutions. Seven sites participated in the experiment by donating a total number of 195 CPUs. Other centers

[4]http://www.irisgrid.es
[5]http://www.gridway.org

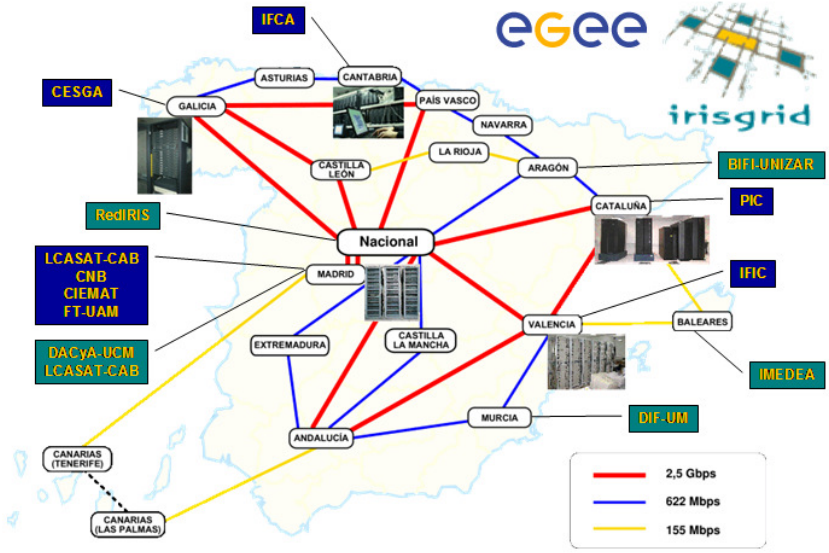

FIG. 2.1. *Geographical distribution and interconnection network of sites.*

participate in the EGEE project, which is composed of more than 100 contracting and non-contracting partners. Seven spanish centers participated by donating a total number of 333 CPUs.

Together, the testbed is composed of 13 sites (note that LCASAT-CAB is both in IRISGrid and EGEE) and 528 CPUs. In the experiments below, we limited to four the number of jobs simultaneously submitted to the same resource, with the aim of not saturating the whole testbed, so only 64 CPUs were used at the same time. All sites are connected by RedIRIS, the Spanish Research and Academic Network. The geographical location and interconnection links of the different sites are shown in Figure 2.1.

3. Core Grid Middleware: Globus. Globus services allow secure and transparent access to resources across multiple administrative domains, and serve as building blocks to implement the stages of Grid scheduling [14]. Table 3.1 summarizes the core Grid middleware components existing in both IRISGrid and EGEE resources used in the experiments. In the case of EGEE, we only used Globus basic services, ignoring any higher-level services, like the resource broker or the replica location service.

TABLE 3.1
Core Grid middleware.

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

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

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

In a Globus-based environment, the user is responsible for manually performing all the submission steps [14] in order to achieve any functionality. To overcome this limitation, GridWay [11] was designed with a *submit & forget* philosophy in mind. The core of the GridWay framework is a personal *submission agent* that performs all scheduling stages and watches over the correct and efficient execution of jobs on Globus-based Grids. The GridWay framework provides adaptive scheduling and execution, as well as fault tolerance capabilities to handle the dynamic Grid characteristics.

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

1. *prolog*: It prepares the remote system by creating a experiment directory and transferring the input files from the client.
2. *wrapper*: It executes the actual job and obtains its exit status code.
3. *epilog*: It finalizes the remote system by transferring the output files back to the client and cleaning up the experiment directory.

This way, GridWay 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, GridWay doesn't require any middleware installation nor network connectivity in the compute nodes.

Other projects [5, 6, 8, 16] have also addressed resource selection, data management, and execution adaptation. We do not claim innovation in these areas, but remark the advantages of our modular, decentralized and "end-to-end" architecture for job adaptation to a dynamic environment.

In this case, we have taken full advantage of the modular architecture of GridWay, as we didn't have to directly modify the source code of the *submission agent*. We extended the *resource selector* in order to understand the GLUE schema used in EGEE. The *wrapper* module also had to be modified in order to perform an explicit file staging between the front-end and the compute nodes in EGEE clusters.

5. Grid Application: Computational Proteomics. 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. In the following experiments, we will consider a Bioinformatics application aimed at predicting the structure and thermodynamic properties of a target protein from its amino acid sequence [10].

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

To speed up the analysis and reduce the data needed, the PDB files are pre-processed 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. The biological results of the comparative study of several families of orthologous proteins are presented elsewhere [4].

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

FIG. 6.1. *Testbed dynamic throughput during the five experiments and theoretical throughput of the most powerful site.*

The experiment files consists 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 in different days during a week. The average turnaround time for the five experiments was 43.37 minutes.

Figure 6.1 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 that the number of active jobs per resource was limited to four). The throughput achieved on each experiment varies considerably, due to the dynamic availability and load of the testbed. For example, resource ce00 at site LCASAT-CAB was not available during the execution of the first experiment. Moreover, fluctuations in the load of network links and computational resources induced by non-Grid users affected to a lesser extent in the second experiment, as it was performed at midnight.

7. Grid Benchmarking Model. In this section we apply a benchmarking methodology to analyze the performance of computational Grid infrastructures in the execution of a High Throughput Computing (HTC) applications [12]. In order to assess this model we will use the execution of the Bioinformatics application explained in previous sections. The benchmarking process used here provides a way to investigate performance properties of Grid environments, to predict the performance of this category of applications, and compare different platforms by inserting performance models in the benchmarking process.

7.1. Workload Characterization. In order to obtain the workload characterization of a grid, we have considered the formerly mentioned Bioinformatics application, that comprises the execution of a set of independent tasks, each of which performs the same calculation over a subset of parameter values. In the execution of this kind of applications, a grid can be considered, from the computational point of view, as an array of *heterogeneous* processors. Therefore, the number of tasks completed as function of time is given by the following equation:

$$n(t) = \sum_{i \in G} N_i \left\lfloor \frac{t}{T_i} \right\rfloor \tag{7.1}$$

where N_i is the number of processors in the Grid (G) that can compute a task in T_i seconds.

The best characterization of the Grid can be obtained if we take the line that represents the average behavior of the system. The next formula represents this line using the r_∞ and $n_{1/2}$ parameters defined by Hockney and Jesshope [9]:

$$n(t) = r_\infty t - n_{1/2} \text{ with } m = r_\infty \text{ and } b = -n_{1/2} \tag{7.2}$$

These parameters are called:

- Asymptotic performance (r_∞): the maximum rate of performance in task executed per second. In the case of an homogeneous array of N processors with an execution time per task T, we have $r_\infty = N/T$.
- Half-performance length ($n_{1/2}$): the number of task required to obtain the half of asymptotic performance. This parameter is also a measure of the amount of parallelism in the system as seen by the application. In the homogeneous case, for an embarrassingly distributed application we obtain $n_{1/2} = N/2$.

The following equation defines the performance of the system (tasks completed per time) on actual applications with a finite number of tasks based on the linear relation of Eq. 7.2:

$$r(n) = n(t)/t = \frac{r_\infty}{1 + n_{1/2}/n} \tag{7.3}$$

The half-performance length ($n_{1/2}$) provides a quantitative measure of the homogeneity in a grid. We can define the degree of homogeneity (v) as

$$v = \frac{2n_{1/2}}{N}. \tag{7.4}$$

This parameter varies from $v = 1$ in the homogeneous case, to $v \approx 0$ when the actual number of processors in the Grid is much greater than the apparent number of processors (highly heterogeneous).

7.2. Benchmarking Methodologies. Previously, we have defined the r_∞ and $n_{1/2}$ parameters to obtain a Grid model. These parameters can be determined in two ways:

- *Intrusive* benchmarking. The system parameters are calculated by linear fitting to the experimental results obtained in the execution of large-scale HTC applications. In order to empirically determine r_∞ and $n_{1/2}$ the benchmarking process should be intrusive to exercise all the resources in the testbed ($n >> N$).
- *Non-intrusive* benchmarking. In general, it may not be feasible to run such an intrusive high throughput benchmark for large Grid environments. In this situation, the r_∞ and $n_{1/2}$ parameters can be computed using 7.3 and raw performance data (average wall time per task, T_i) of each resource. Let us consider $T_i = T_i^{xfr} + T_i^{exe} + T_i^{sch}$, where T_i^{xfr} and T_i^{exe} are the average file transfer and execution times in host i; and T_i^{sch} is the scheduling time which represents the resource selection overhead. We can rewrite Eq. 7.1 as:

$$n(t) = \sum_{i \in G} N_i \left\lfloor \frac{t}{T_i^{xfr} + T_i^{exe} + T_i^{sch}} \right\rfloor. \tag{7.5}$$

This equation can be used to obtain the *non-intrusive* r_∞ and $n_{1/2}$ parameters by fitting the best straight line. We can also estimate the influence of the resource selection overhead by comparing the non-intrusive r_∞ and $n_{1/2}$, with those obtained by setting T^{sch} to 0 in Eq. 7.5.

7.3. Experiments and Results. We begin the analysis presenting the intrusive and non-intrusive measurement made on testbed of the parameters r_∞ and $n_{1/2}$. Figure 7.1 shows the experimental performance obtained in the first two executions of the mentioned Bioinfomatics application, along with that predicted by Eq. 7.3. The r_∞ and $n_{1/2}$ have been calculated by linear fitting to: (i) experimental results obtained in the execution of the application; (ii) Eq. 7.5 using the average file transfer and execution times of each host and $T^{sch} = 60s$; and also (iii) Eq. 7.5 with $T^{sch} = 0s$.

This figure shows the effects of the resource selection on the optimum performance. The resource selection process reduces the asymptotic performance of the Grid, because of a delay between different tasks submitted

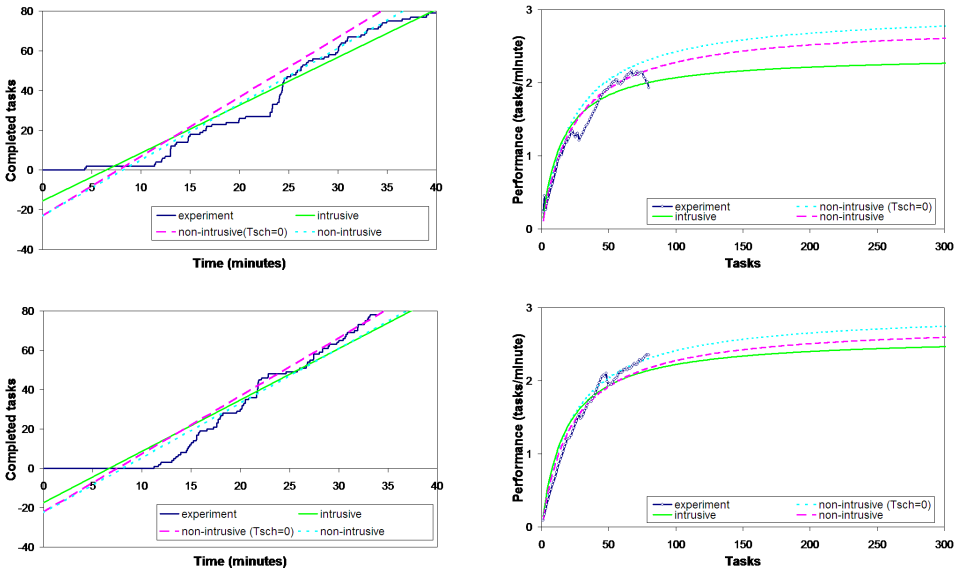

FIG. 7.1. *Measurements of r_∞ and $n_{1/2}$ on the testbed based on experimental data, and raw resource performance (left-hand charts). Experimental performance of the Bioinformatic application, along with that predicted by Eq. 7.3 (right-hand charts).*

to the same host. This delay is mainly due to the Globus MDS update frequency and the Grid*W*ay resource broker. However, it does not affect to the $n_{1/2}$ parameter, since the brokering overhead increases the execution time (T_i) by the same amount in all the resources.

Table 7.1 shows the values of the r_∞ and $n_{1/2}$ coefficients for each execution of the experiments. For the shake of the completeness, we also include the turnaround time (T_{Grid}). Based on the *intrusive* results, the testbed is characterized by an average asymptotic performance of 2.31 tasks per minute. Furthermore, in order to achieve the half of this asymptotic performance it is necessary to execute, on average, 15.34 tasks. And so, the apparent number of resources to the application is approximately 30.68, with an execution time of 13.26 minutes per task.

In Table 7.1, we can also see the degree of homogeneity (v) for the intrusive and non-intrusive case. Thus, the average of the intrusive v is 0.61 (low homogeneity degree), and the average of non-intrusive v is 0.9, closer to the homogeneous case ($v = 1$). This difference is mainly due to the experiments not saturating the testbed, as can be seen in the right-hand charts of Figure 7.1. Moreover, this difference is also consequence of calculating non-intrusive v by applying the model under ideal conditions, therefore yielding in an homogenization of the results.

TABLE 7.1
Turnaround time T_{Grid}, $r_\infty(tasks/minute)$, $n_{1/2}$ (tasks), and degree of heterogeneity coefficient v for each experiment.

	Intrusive			Non-Intrusive			
Experiment	r_∞	$n_{1/2}$	v	r_∞	$n_{1/2}$	v	T_{Grid}
1	2.38	15.02	0.58	2.85	23.20	0.89	42.00
2	2.61	17.48	0.70	2.79	22.54	0.90	33.82
3	2.04	17.04	0.71	2.74	21.52	0.90	46.05
4	2.14	13.88	0.51	2.67	23.64	0.88	50.37
5	2.40	13.27	0.53	2.95	22.73	0.91	44.87

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

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, GridWay, can work with Globus, as a standard core Grid middleware, over any Grid fabric in a *loosely-coupled* way. The smooth process of integration of two so different testbeds, although both are based on Globus, demonstrates that the GridWay approach (i. e. the Grid way), based on a modular, decentralized and "end-to-end" architecture, is appropriate for the Grid. Moreover, the Grid performance model for High Throughput Computing Applications validates the experimental results obtained.

Acknowledgments. This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE) through BIOGRIDNET Research Program S-0505/TIC/000101, by BSCH/UCM through research grant PR27/05-14035-BSCH and by Ministerio de Ciencia y Tecnología, through the research grant TIC 2003-01321. The authors participate in the EGEE project.

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.

REFERENCES

[1] R. J. Allan, J. Gordon, A. McNab, S. Newhouse, and M. Parker, *Building Overlapping Grids*, tech. rep., University of Cambridge, Oct. 2003.

[2] E. B. Carpenter, *RFC 1958: Architectural Principles of the Internet*, June 1996.

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

[4] U. Bastolla, A. Moya, E. Viguera, and R. van Ham, *Genomic Determinants of Protein Folding Thermodynamics in Prokaryotic Organisms*, Journal of Molecular Biology, 343 (2004), pp. 1451–1466.

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

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

[7] I. Foster, *What Is the Grid? A Three Point Checklist*, GRIDtoday, 1 (2002). `http://www.gridtoday.com/02/0722/100136.html`

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

[9] R. Hockney and C. Jesshope, *Parallel Computers 2:Architecture Programming and Algorithms*, Adam Hilgee Ltd., 1998.

[10] E. Huedo, U. Bastolla, R. S. Montero, and I. M. Llorente, *A Framework for Protein Structure Prediction on the Grid*, New Generation Computing, 23 (2005), pp. 277–290.

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

[12] R. S. Montero, E. Huedo, and I. M. Llorente, *Benchmarking of High Ttroughput Computing Applications on Grids*, Parallel Computing, (2006). in press.

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

[14] J. M. Schopf, *Ten Actions when Superscheduling*, Tech. Rep. GFD-I.4, Scheduling Working Group—The Global Grid Forum, 2001.

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

[16] S. Vadhiyar and 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.