J Grid Computing
DOI 10.1007/s10723-010-9177-5

Using a Simple Prioritisation Mechanism to Effectively Interoperate Service and Opportunistic Grids in the EELA-2 e-Infrastructure

Francisco Brasileiro · Matheus Gaudencio · Rafael Silva · Alexandre Duarte · Diego Carvalho · Diego Scardaci · Leandro Ciuffo · Rafael Mayo · Herbert Hoeger · Michael Stanton · Raul Ramos · Roberto Barbera · Bernard Marechal · Philippe Gavillet

Received: 29 June 2010 / Accepted: 30 November 2010

Abstract Grids currently in production can be broadly classified as either service Grids, composed of dedicated resources, or opportunistic Grids that harvest the computing power of non-dedicated resources when they are idle. While a service Grid provides high and well defined

F. Brasileiro (✉) · M. Gaudencio · R. Silva · A. Duarte
Universidade Federal de Campina Grande, Av. Aprígio Veloso, s/n, 58.429-900, Campina Grande, PB, Brazil
e-mail: fubica@dsc.ufcg.edu.br

M. Gaudencio
e-mail: matheusgr@lsd.ufcg.edu.br

R. Silva
e-mail: rafael@lsd.ufcg.edu.br

A. Duarte
e-mail: alex@dsc.ufcg.edu.br

D. Carvalho
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, Av. Maracanã 229, 20271-110, Rio de Janeiro, RJ, Brazil
e-mail: d.carvalho@ieee.org

H. Hoeger
Universidad de Los Andes, CeCalCULA, Av. 4, Edif. General Masini, Piso 3, Mérida, 5101, Venezuela
e-mail: hhoeger@ula.ve

L. Ciuffo · M. Stanton
Rede Nacional de Ensino e Pesquisa, R. Lauro Müller 116 / 1103, 22290-906, Rio de Janeiro, RJ, Brazil
e-mail: leandro.ciuffo@rnp.br

M. Stanton
Universidade Federal Fluminende, R. Lauro Müller 116 / 1103, 22290-906, Rio de Janeiro, RJ, Brazil
e-mail: michael@rnp.br

D. Scardaci
Istituto Nazionale di Fisica Nucleare, Sezione di Catania, Via S. Sofia, 64, 95123, Catania, Italy
e-mail: diego.scardaci@ct.infn.it

R. Barbera
Dipartimento di Fisica e Astronomia dell'Universitá di Catania, Via S. Sofia, 64, 95123, Catania, Italy

R. Barbera
INFN, Via S. Sofia, 64, 95123, Catania, Italy
e-mail: roberto.barbera@ct.infn.it

R. Mayo
CIEMAT, Avda. Complutense, 22, 28040 Madrid, Spain
e-mail: rafael.mayo@ciemat.es

R. Ramos · B. Marechal · P. Gavillet
CETA-CIEMAT, C/ Sola, 1, 10200 Trujillo, Spain
e-mail: raul.ramos@ciemat.es

B. Marechal · P. Gavillet
CERN, C/ Sola, 1, 10200 Trujillo, Spain

B. Marechal
e-mail: marechal@if.ufrj.br

P. Gavillet
e-mail: philippe.gavillet@cern.ch

Published online: 12 January 2011

levels of quality of service, an opportunistic Grid provides only a best-effort service. Nevertheless, since opportunistic Grids do not require resources to be fully dedicated to the Grid, they have the potential to assemble a much larger number of resources. Moreover, these Grids cater very well to the execution of the so-called embarrassingly parallel applications, a type of application that is frequently found in practice, and that comprises the largest portion of the typical workload processed in production Grid systems. The EELA-2 e-infrastructure is comprised of a service Grid and an opportunistic Grid that federates computing resources from scientific institutions in both Europe and Latin America. Due to the complementary characteristics of these two types of Grids, a lot of attention has recently been placed in how to interoperate them. In this paper we focus on the less studied problem of assessing the feasibility of such interoperation. We analyse different prioritisation policies that define when the resources of one Grid can be used to run jobs originating from the other. Our results show that in the absence of a suitable prioritisation policy, the benefits that the users of one Grid may have, frequently come with an important negative impact on the users of the other Grid. We also show that a simple reciprocation mechanism is capable of arbitrating the interoperation in such a way that, whenever possible, users profit from the interoperation and, in no case, this benefit leads to a noticeable reduction on the quality of service that the users would experience were the Grids not to interoperate. We conclude discussing how we have implemented, in the context of the EELA-2 project, this prioritisation mechanism, allowing the effective interoperation of a service Grid based on the gLite middleware with an opportunistic Grid that uses the OurGrid middleware.

Keywords Grid computing · Service Grids · Opportunistic Grids · Prioritisation · EELA-2

1 Introduction

Throughout this decade, Grid computing has established itself as a key technology for the support of an important portion of the scientific activity developed worldwide, a trend that has been dubbed *e-Science*. As the technology matured, several Grid infrastructures, or *e-infrastructures* for short, have been built and are currently in operation. In particular, the authors have been deeply involved in initiatives to deploy and operate an e-infrastructure gathering resources from research centres in several Latin American and European countries. These efforts were initiated in 2006 in the context of the "E-infrastructure shared between Europe and Latin America" (EELA) project[1] [19]. The EELA project was a 2-year project run by 21 institutions from Europe and Latin America under the 6th Framework Programme for Research, Technological Development and Demonstration (FP6) of the European Commission (EC). The objective of the EELA project was to bring the e-infrastructures of Latin American countries to the level of those already available in Europe. For this purpose it built a service Grid pilot infrastructure for scientific applications—benefiting from the mature state of the EC funded project "América Latina Interconectada Con Europa" (ALICE)[2] and of its product, the RedCLARA network—with the ultimate goal of promoting a sustainable framework for e-Science in Latin America. This e-infrastructure was supported by the gLite middleware.[3]

Service Grids normally assemble high performance, dedicated computing resources, such as clusters, supercomputers, and large data storage systems that are spread over a relatively small number of administrative domains. For instance, the largest service Grid currently in operation is the one created in the context of the EGEE series of projects[4] [14] which, at the time of writing, encompassed more than 160,000 processing cores distributed among approximately 270 resource centres in more than 60 countries.[5] Service Grids provide high and well defined levels of quality of

[1] http://www.eu-eela.org/first-phase.php.

[2] http://alice.dante.net/.

[3] http://cern.ch/glite.

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

[5] http://gstat-prod.cern.ch/gstat.

service (QoS). To achieve such levels of QoS, a lot of effort is employed in the monitoring and management of the infrastructure. Moreover, these infrastructures are set up to run complex distributed applications that, in many cases, require important core Grid services to be in operation.

The EELA project was very successful, and allowed small and medium research groups in Latin America, that use to work mostly isolated, to take part in important global research projects [15, 20]. Also, the initiative was important to foster the development of new collaborations among the research groups involved, in the best spirit of the e-Science trend. Despite its many success stories, the development of the EELA project has also highlighted problems that called for a redesign of the architecture of the e-infrastructure built. From the resource provision viewpoint, many institutions, notedly in Latin America, found it difficult to commit dedicated resources to the service Grid. The overwhelming majority of research groups in Latin America, as in many other parts of the world, are small, counting on a dozen of people or so. Even in the case when they have the appropriate computing resources to be incorporated in the service Grid, they lack the skilled computing support team to install their resource centres and, most importantly, to maintain them operating with the required QoS level. Nevertheless, it was possible to identify many partners that could provide a relatively large amount of non-dedicated resources to the infrastructure in a best-effort basis, which lead us to consider the option of deploying an alternative Grid system that could use these resources opportunistically.

Opportunistic Grids are somewhat more “lightweight” Grid infrastructures that scavenge idle computing cycles from non-dedicated resources. Several types of opportunistic Grids have been proposed, implemented and deployed. Among the most important representatives of this class of Grids one can list: *desktop* Grids, first proposed by the Condor project [9, 18]; *voluntary computing* platforms such as the pioneer SETI@home system [1] and its successor BOINC [2] and, more recently, *peer-to-peer* (P2P) Grids such as those supported by the OurGrid middleware [4, 6]. In all cases, the functioning of an opportunistic Grid is very similar and simple. The owners of the computing resources install a Grid agent that is in charge of monitoring the utilisation of the resource and detect when it is idle. Upon detecting that the resource is idle, the agent passes this information to a scheduler that is in charge of dispatching tasks to be executed on the idle resource. Whenever a resource is required to run applications spawned by its owner, immediately, any Grid-related task that may be running is either suspended or interrupted.

Since the availability of the resources in opportunistic Grids depends on the behavioural pattern of the resource owners, it is normally not possible to ensure any QoS guarantees in this type of Grid. Thus, they work basically on a best-effort basis. Nevertheless, this relatively unsophisticated strategy has proved to be very effective in assembling enormous amounts of computing cycles for the execution of scientific applications [10]. Moreover, embarrassingly parallel—also known as *bag-of-tasks* (BoT)—applications, i.e. those parallel applications that can be divided in a large number of independent sequential tasks that do not communicate with each other, can be very easily scheduled and efficiently executed in this kind of infrastructure [7].

Given the simplicity of most opportunistic Grid middleware—compared to their service Grid counterparts—setting up an opportunistic Grid seemed to be a better option for many of the EELA partners. Also, from the resource consumption viewpoint, although a service Grid is very flexible in the types of applications that it can support, the learning curve required to successfully port the application in these complex infrastructures may turn out to be an insurmountable hurdle for many small research groups. Again, opportunistic Grids seemed to be a better fit for these users.

In summary, service Grids and opportunistic Grids are very different systems, catering to different needs, and providing diverse facilities to their users and different challenges to their administrators. On one side, service Grids gather dedicated clusters and supercomputers, in a relatively small number of sites, to offer high levels of QoS to all sorts of distributed and parallel applications, at the expenses of a high administrative cost. On the other side, opportunistic Grids scavenges the idle cycles of desktop machines, spread over many sites, providing a best-effort service to embarrassingly parallel applications, at a reasonably low cost.

In April 2008, shortly after the conclusion of the EELA project, most of the partners involved in this project started another 2-year project, called “E-science Grid facility for Europe and Latin America” (EELA-2).[6] The EELA-2 project was also an initiative co-funded by the EC within its 7th Framework Programme for Research, Technological Development and Demonstration (FP7), and involving 78 institutions from 11 countries in Latin America and 5 in Europe. The EELA-2 project aimed at transforming the pilot infrastructure put in place by its predecessor project into a high capacity, production-quality, scalable e-infrastructure, providing round-the-clock worldwide access to distributed computing, storage and network resources for a wide spectrum of applications from European and Latin American scientific communities. In addition to the service Grid, the approach followed also included the deployment of an opportunistic Grid infrastructure supported by the OurGrid middleware [6]. Moreover, the existence of two different infrastructures under the same governance, motivated us to seek alternatives for allowing the interoperation between the service Grid and the opportunistic Grid that together comprised the EELA-2 e-infrastructure.

Despite their differences, a great deal of attention has recently been placed in devising ways to allow service and opportunistic Grids to interoperate. This trend has been triggered by many catalysers, from which the most common is the desire to increase the capacity of each individual Grid with the resources of the other Grid. Other motivations include the need to better cater to a more diverse set of user requirements and to make the operation of the federated Grids more cost-effective. All three motivations were present in the EELA-2 context.

Most of the research in this area has been devoted to understand which are the best ways to accomplish interoperation [11, 13, 21, 22, 24]. The gateway approach is one of the most well-known alternatives to achieve this end [11, 24, 27]. In this approach, a gateway element is deployed to bridge the two systems. It routes jobs submitted in one infrastructure to the other, possibly translating the job written in the language accepted by one middleware into a job that is accepted by the other middleware.

Although significant progress has been made in the development of such gateways, there has been little work on how to define policies to govern the interoperation of two independent Grid infrastructures bridged by them. Note that the interests of the providers and users of the two parts of the infrastructure are not necessarily aligned. Taking the most common reason for interoperating two Grid systems into account, at first sight, one may argue that if both Grids are either under-provisioned or over-provisioned, then there is little incentive to interoperate, since in the first case neither of the Grids have excess resources to provide to the other, while in the second case, extra resources are not needed. Similarly, when one system is under-provisioned and the other is over-provisioned, then the benefits of interoperation will only be seen by one of the systems. Again, at least for the over-provisioned Grid, there is no interest for interoperation. However, Grid workloads are usually highly variable in time, with periods of very high load when all resources are busy and some jobs may need to be queued waiting for available resources, and periods of lower load when some of the resources sit idle. Thus, if we consider the workload of the independent Grid systems, it is possible that, over time, there will be periods when it is interesting for both of them to interoperate, periods when only one of them can profit from the interoperation with the other, and periods when interoperation is not beneficial to either of them.

From the above, it is clear that, unless a suitable arbitration mechanisms is in place, it is not possible to guarantee that the QoS experienced by two Grid systems that interoperate is never worse than the QoS that each of them would experience were they operated independently. In this paper we investigate this issue. Our methodology is based on the use of simulations fed with data from workloads of a production Grid. We show

[6]http://www.eu-eela.eu/.

that in the absence of a prioritisation mechanism it is possible that one Grid exploits the other in an unfair way. Then, we show how the use of the simple reciprocation mechanism proposed by Andrade et al. [3] can be used to arbitrate the interoperation of a service Grid and an opportunistic Grid in a very effective way. Finally, we discuss how we have implemented this approach in the EELA-2 e-infrastructure to allow the arbitrated interoperation of its service Grid and its opportunistic Grid.

The rest of this paper is organised as follows. In Section 2 we present the system under study, which includes the simulation model for the service and the opportunistic Grids, the workload that was used in the simulation experiments that are discussed throughout the paper, as well as the Grid resources considered. For the sake of reproducibility of our results, all data used in the simulations performed, as well as the source code and executables of the simulators used are available online at http://redmine.lsd.ufcg.edu.br/wiki/ourgridg3/Grid_Simulator. The assessment of the effectiveness of interoperating the Grids without arbitration is presented in Section 3, while Section 4 is devoted to discuss the impact of arbitration. Our implementation of the reciprocation mechanism evaluated is presented in Section 5. We survey works that are related to ours in Section 6. Finally, our concluding remarks and directions for future work are presented in Section 7.

2 System Description

2.1 Simulation Model

The simulation model represents two Grid systems that can operate both separately and in cooperation. One of the Grids is a service Grid, while the other is an opportunistic one. The service Grid is composed of resource centres, each of which possessing a single cluster with a number of nodes. The opportunistic Grid, on the other hand, is composed of independent sites, each of which providing a number of desktops to the Grid. Each resource centre (resp. site) of the service Grid (resp. opportunistic Grid) is an independent administrative domain.

When the two Grids interoperate, then jobs are submitted from one Grid to the other through one or more gateways. There are different ways to implement these gateways. In this section we will only describe how the Grids behave when they are working independently. We leave the description of how they can interoperate to next two sections.

Each Grid has its own workload. There are two types of jobs that may be part of the Grid workload, namely: sequential and parallel jobs. Sequential jobs are run on a single cluster node or desktop, while parallel jobs are run on two or more nodes in the same cluster. A service Grid usually federates clusters, thus, its workload is formed by both types of jobs. On the other hand, an opportunistic Grid usually federates desktops that are not suitable to execute most parallel jobs. Thus, its workload is formed only by sequential jobs. Sequential jobs that are submitted by the same user at the same time are grouped in a single BoT job. We refer to each sequential job in such a group as a task of the BoT job. For simplicity, all sequential jobs are treated as BoT jobs, some of them with just a single task.

The Grid schedulers are in charge of allocating a suitable cluster or desktop to run each job of the workload that is submitted to them. We assume that their objective is to minimise the mean makespan of the jobs they handle, where the job makespan is defined as the time comprised between the job submission and the job completion. Note that for BoT jobs, the time of completion of the job is given by the completion time of the task that finishes last.

The scheduling algorithm of the service Grid works as follows. Jobs submitted are handled by a central *Workload Management System* (WMS). We assume that the central WMS is fed with perfect information and is able to accurately identify which is the cluster that will be able to provide the smallest makespan for each job submitted. (Note that we are not interested in the absolute performance of the system, but rather on how its performance is affected by the fact that it interoperates with another Grid. Thus, this assumption tries to reduce the sources of non-determinism in the outcome of the scheduling.) The job is then sent to this cluster. We assume that the scheduler of the cluster follows a simple *first-come-first-served* (FCFS) policy. Thus, jobs sent to a particular cluster are always placed in the end of the cluster's queue. The job that is in the first position of the queue starts its execution as soon as the number of idle nodes in the cluster is larger than the number of nodes required by this job.

Each site of the opportunistic Grid runs its own scheduler that is in charge of scheduling the jobs originating from its associated site. From the perspective of the site scheduler, jobs originating from its site are called *local* jobs, while jobs that come from the other sites are referred to as *remote* jobs. Similarly, desktops from its site are called *local* resources, while desktops from the other sites are called *remote* resources.

The scheduling algorithm is very simple. Whenever a BoT job is submitted, the scheduler tries to schedule each of its tasks in an idle resource. Firstly, the scheduler finds out if there is a local resource available. If not, the scheduler checks if there are local resources executing tasks of remote jobs. In this case, one of the tasks of remote jobs is pre-empted and the local resource in which it executed is allocated to one of the tasks of the local job. If no local resources are available, then the scheduler tries to find out remote resources that could run the tasks of its job. Remote resources are searched by randomly querying the other sites one after the other. If no available resources are found, the scheduler waits until any of the tasks running in the Grid finishes and starts the process again. In a real system one should not expect that task completion notifications are sent to all sites whenever a task is finished; thus, scheduling retries should be time-driven instead of event-driven. Again we took this simplifying assumption to reduce the sources of non-determinism.

In this paper we consider that the provision of local desktops to run tasks of remote jobs can be done in two ways. In the first model we use a history-less equitable sharing policy, i.e. in case the schedulers of two remote sites contend for the same desktop, then the desktop is allocated to run the task coming from the site that is currently using less desktops from the local site. When necessary, a random selection is performed to break ties.

We have also modelled a more sophisticated sharing policy that is based on a reciprocation mechanism named the "Network of Favours" (or NoF, for short) [3]. The NoF policy is suitable to be used in P2P systems because it promotes contributions and marginalises sites that do not contribute with their idle resources to the Grid (the latter are referred in the literature as *free-riders*). In a NoF-based system, each site is represented by a *peer agent* (or a peer, for short) that is in charge of maintaining a balance of the pairwise past interactions that it has had with other peers in the system. Let $B_p(q)$ be the balance that peer p maintains for peer q. $B_p(q)$ is computed in such a way that it is never smaller than zero, and is an indication of the amount of computing power that p owns to q. Whenever two or more peers contend to get resources from p, then p donates its resources to whoever peer it owns most. Again, if necessary, a random selection is used as a tie-breaker. Not allowing the balance to go negative is required to avoid whitewash attacks [3]. The correct calculation of the balance is also a challenging issue; the interested reader should refer to the work by Santos et al. [23] for details.

In the opportunistic Grid, tasks can be pre-empted. This can happen both when the resource is reclaimed by its owner (for instance, when the user of a desktop starts a non-Grid application), as well as when a task of a local job requires the local resource that is executing a task of a remote job, and also when the sharing policy decides that the resource that is running a task of remote job should be allocated to run the task of another remote job. For the sake of simplicity, we assume that the tasks that are executed in the opportunistic Grid are able to perform checkpoints, so that when they are pre-empted, they can later resume their execution, in the same or in another resource, from the point where they have been pre-empted. This simplification should not impact too much the conclusions that we can draw from our experiments, since, as stated before, we are not concerned with absolute performance.

2.2 Workload Description

In this work, the simulations were performed using workloads from real systems available at the Grid Workloads Archive maintained by the University of Delft in the Netherlands [16]. From the several workloads available, we selected the NorduGrid[7] one, since it had the largest number of sites (75) and users (387), as well as a substantial number of jobs (781,370).

As it is described in the GWA site, "*NorduGrid is a production Grid for academic researchers in Denmark, Estonia, Finland, Norway, Sweden, etc. Since 2002, NorduGrid has been in continuous operation and development, and since 2003 industrial, scientific or private organisations with interest in Grid computing have been invited to contribute their compute power to the NorduGrid as collaborators. In NorduGrid, non-dedicated resources are connected using the Advanced Resource Connector (ARC) as Grid middleware.*"

The trace comprises data about jobs submitted from March 2003 until May 2006. It contains all types of jobs previously mentioned, with a majority of them (91.45%) falling in the BoT class. It is important to point out that the trace only explicitly differentiates parallel and sequential jobs, as it stores the number of processors that have been used to run each job. BoT jobs were inferred by inspecting the submission times of the jobs. We considered that two jobs submitted by the same user were part of the same BoT job if it was unlikely that the outcome of the first job submitted had triggered the execution of the other job. Given the mean duration time of the jobs available in the workload, we considered that two jobs from the same user that were submitted within less than 120 s apart belonged to the same BoT job.

To restrict the time required to run the simulations we have divided the whole workload in chunks of 4,000 "jobs" each, as per the definition of jobs in GWA. Note that, as mentioned before, since the trace does not differentiates tasks of a BoT job from sequential or parallel jobs, a chunk usually has less than 4,000 jobs, as per the definition we use in this paper. We then selected some of these chunks to use as input data to our simulations. We divided the chunks in those that contained all types of jobs and those that contained only BoT jobs. The former were used as input to the service Grid, while the latter were used as the input to the opportunistic Grid. In addition to the type of jobs that they contained, chunks were also selected based on their load characteristics and the number of different sites that they contained. We selected 66 chunks for the service Grid, each with 32 resource centres, and 36 chunks for the opportunistic Grid, each with 24 sites. For each type of Grid, half of the chunks represented periods of high contention for Grid resources, while the other half represented low contention scenarios.

We used the following information from the trace:

- *JobId*: represents the job identification number;
- *SubmitTime*: specifies the time at which the job was submitted; since we have used chunks of the trace, for each chunk, the submission time of a particular job is given by the difference between its absolute submission time and the absolute submission time of the first job in the chunk;
- *RunTime*: specifies the wall clock time (in seconds) that the execution of the job took, i.e., the difference between the end time and start time;
- *NProc*: represents the number of processors required by the job; we assume that parallel jobs are those with $NProc > 1$;
- *OriginSiteID*: identifies the Grid site from which the job originates.

When jobs are submitted to clusters, they need to specify both the number of processors required, as well as for how long these nodes should be allocated for the job. This information is missing in many on the workload entries we used. Therefore, we took again a simplifying approach to assume that users' prediction of the running time of their jobs was perfect and made the requested cluster time to be exactly the time elapsed to process the job.

2.3 Grid Resources

In our simulations, the service Grid comprises 480 nodes in total. To allow for the execution of all

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

parallel jobs found in the trace, one of the resource centres has the number of nodes required by the job that requires more processors to run (27) and the remaining nodes are evenly spread over the other 31 resource centres.

The opportunistic Grid has two different configurations that are used to control the contention level in the Grid. The high contention workload is executed in a Grid composed of 24 sites, each with 10 desktops available, while the low contention workload is executed in a Grid with the same 24 sites, but with each site having 20 desktops available.

Since we are assuming that the jobs that run in the opportunistic Grid are able to checkpoint their execution, the fact that the resources of the opportunistic Grid are shared and may be reclaimed at any time by the resource owner has an impact only on the aggregated computing power that the Grid can provide over some period of time. Therefore, we do not model resource availability in the opportunistic Grid and assume that pre-emption comes only from the prioritisation mechanism implemented by the opportunistic Grid scheduler that runs at each site.

Moreover, we do not model any additional overhead that may be imposed due to job migration in case of pre-emption. Analysing the logs of our simulations we identified that even in the most challenging scenarios (with high contention in both Grids), around 3/4 of the jobs executed without any pre-emption and only 3.3% of the jobs were pre-empted more than 5 times.

Finally, to speed up the execution of the simulator, we assume that the nodes of the service Grid clusters and the desktops of the opportunistic Grid have all the same processing power.

3 Interoperation Without Arbitration

In this section we start to assess the impact that interoperation may have in the QoS perceived by the users of two Grids that work independently. Our first step is to establish a baseline for comparison. Then, we evaluate the impact of having just one Grid exploiting the other. Finally, we measure the impact of having interoperation in both directions. In all interoperation cases, we assume that there is no extra prioritisation mechanisms, except those already present in the independent Grids.

Throughout this paper we will present results that are mean values of the makespan metric over the several jobs that comprise a given workload chunk and over the various workload chunks used in different simulation instances. In all cases, we have run enough simulations to have a confidence level of 95% and an error that is always smaller than 5%.

3.1 Baseline

Our baseline values are derived from the simulations that considered each Grid independently. We have simulated 66 workload chunks for the service Grid and 36 for the opportunistic Grid and have measured the makespan of each of the jobs submitted. The mean makespan of the jobs in all chunks considered is presented in Table 1.

As it can be seen, when we consider the same level of contention, the service Grid has a more demanding workload than the opportunistic Grid for the workload we have used. Moreover, in both Grids, the difference between the mean makespan for the low and the high contention cases are substantially different.

Also, from the point of view of the mean makespan of the Grid, there is no difference between the two sharing policies used in the opportunistic Grid. Therefore, from now on we will only consider the NoF policy when simulating the opportunistic Grid.

Table 1 Mean makespan for baseline scenario: Grids working independently

Grid type	Contention	Mean makespan (s)
Service	Low	67,882
Service	High	154,719
Opportunistic with equitable	Low	36,175
Opportunistic with equitable	High	66,073
Opportunistic with NoF	Low	35,197
Opportunistic with NoF	High	66,559

3.2 Exploiting the Opportunistic Grid

We now evaluate the effect of having interoperation only in the direction of the service Grid to the opportunistic Grid, i.e. resources from the opportunistic Grid can be used to run jobs originating on the service Grid but jobs originating on the opportunistic Grid cannot run in the resources of the service Grid.

In this case, the simulation model of the opportunistic Grid remains the same and we slightly change the simulation model of the service Grid. First of all, there is a peer agent that is associated to the WMS and that it uses whenever it decides to submit service Grid jobs to the opportunistic Grid. Of course, only BoT service Grid jobs can be submitted to the opportunistic Grid. The way this is accomplished is the following. If there are no clusters where the tasks of such a job could be scheduled without having to be placed in a non-empty queue, then the WMS peer tries to obtain a resource at any of the sites of the opportunistic Grid. If there is such a resource available, then the task is scheduled to run there. Notice that the execution of this task follows the policies established by the opportunistic Grid. In particular, it can be pre-empted. In this case, the WMS may schedule the remaining processing to be executed in the service Grid, if there are resources readily available, or at another resource that it manages to obtain from the opportunistic Grid. Table 2 presents the mean makespan for the simulations considering the one-way interoperation just described.

The relative benefit column shows the ratio between the mean makespan obtained in the baseline simulation and the mean makespan obtained for each corresponding setting. Thus, relative benefits greater than one indicate that the mean makespan has been reduced, while values smaller than one indicate increased mean makespan. As expected, the mean makespan of the service Grid workload has experienced a considerable reduction. Also, this improvement comes with a substantial reduction on the QoS of the opportunistic Grid in all but one scenario (the one in which both Grids are under low contention).

3.3 Exploiting the Service Grid

We now investigate the interoperation in the opposite direction, i.e. the opportunistic Grid uses resources of the service Grid, but the service Grid does not use resources of the opportunistic Grid. In this case, the change in the simulation model is as follows. If a site scheduler tries to allocate a desktop to a local job but there are no local or remote resources available, it tries to schedule it to run in the service Grid. However, this is only achieved it there is a cluster with idle nodes and an empty queue. Note that in this case, the job coming from the opportunistic Grid is never preempted, since the service Grid does not pre-empt jobs. The mean makespan values are presented in Table 3 .

Again, the benefits of the exploitation of resources from the other Grid can be clearly identified. The mean makespan for the opportunistic Grid remains approximately the same for its low contention workload, while it is reduced by a factor of 2/3 in the case of the high contention workload. Likewise, the negative impact on the service Grid is only noticed when the opportunistic Grid is under high contention, being larger for the case in which the service Grid is also under high contention.

Table 2 Mean makespan for one-way interoperability: service Grid exploiting the opportunistic Grid

Contention		Grid type			
Service	Opportunistic	Service		Opportunistic	
		Mean makespan (s)	Relative benefit	Mean makespan (s)	Relative benefit
Low	Low	43,231	1.57	36,466	0.97
Low	High	43,853	1.55	148,624	0.45
High	Low	50,537	3.06	69,906	0.50
High	High	93,903	1.65	148,149	0.45

Table 3 Mean makespan for one-way interoperability: opportunistic Grid exploiting the service Grid

Contention		Grid type			
Service	Opportunistic	Service		Opportunistic	
		Mean makespan (s)	Relative benefit	Mean makespan (s)	Relative benefit
Low	Low	67,843	1.00	35,979	0.98
Low	High	72,073	0.94	43,014	1.55
High	Low	158,643	0.98	35,067	1.00
High	High	214,820	0.72	43,038	1.55

3.4 Simultaneous Exploitation

The two sets of simulations described above have shown that, indeed, for the traces we have studied, both Grids can benefit from the interoperation with the other. However, they also point out that this benefit may come with a negative impact on the other Grid. To complete our analysis, we have executed simulations that mix the two scenarios previously studied. In this setting, we assess the system behaviour with full interoperability between the two Grids under all combinations of workload contention. The mean makespan for these experiments are shown in Table 4.

The results show that, for the workload studied, the service Grid is able to retain all the benefits that we have measured in the previous experiment, while the opportunistic Grid could only retain part of the benefits, and for the case in which it is in low contention and the service Grid is under high contention, its QoS is degraded.

We point out that these results are valid only for the workload we have used. It is possible that for a different workload, other situations would arise, with cases in which only the opportunistic Grid would benefit, both Grid would benefit and neither Grid would benefit from the interoperation. What our results highlight is that, unless some arbitration mechanism is put in place to define which jobs originating from one Grid should be allowed to be executed on the other Grid, it is very difficult to predict beforehand what is the benefit that the interoperation will bring. This in turn is a big disincentive for interoperation to happen.

In the next section we propose a very simple arbitration mechanism that could be used to ensure that the QoS experienced by the users of one Grid is never reduced by the fact that their Grid is able to interoperate with other Grids.

4 A Simple Arbitration Mechanism

The arbitration mechanism described in this section is based on the NoF mechanism proposed for P2P Grids [3]. Under this arbitration mechanism, each resource centre in the service Grid runs a peer agent. Any node in the cluster that is not running a job that came from the cluster's queue is given to the local peer. These idle nodes are then made available to the opportunistic Grid to run BoT jobs originating from both the service Grid and the opportunistic Grid. A cluster node that is being opportunistically used to run one of these jobs is pre-empted whenever there is a job in the cluster's queue that requires it.

Table 4 Mean makespan for full interoperability

Contention		Grid type			
Service	Opportunistic	Service		Opportunistic	
		Mean makespan (s)	Relative benefit	Mean makespan (s)	Relative benefit
Low	Low	43,231	1.57	34,944	1.01
Low	High	44,413	1.53	44,066	1.51
High	Low	52,120	2.97	41,716	0.84
High	High	91,968	1.68	51,585	1.29

Table 5 Mean makespan for full interoperability with arbitration

Contention		Grid type			
Service	Opportunistic	Service		Opportunistic	
		Mean makespan (s)	Relative benefit	Mean makespan (s)	Relative benefit
Low	Low	43,234	1.57	36,332	0.97
Low	High	43,254	1.57	43,241	1.54
High	Low	44,268	3.50	36,955	0.95
High	High	44,595	3.47	45,145	1.47

Now, the submission of jobs from the opportunistic Grid to the service Grid follows the original algorithm presented in Section 2. Note that in this setting, the opportunistic Grid has more resources, since it is able to exploit the dedicated resources of the service Grid in an opportunistic way, whenever they are idle. On the other hand, the submission of service Grid jobs works pretty much in the same way described in Section 3.2, with the only difference that instead of using a central peer associated to the WMS, the WMS uses the peer associated to the resource centre from which the job originates.

We have run simulations for the same workload used in the previous scenarios. Table 5 presents the mean makespan for these simulations.

As it can be seen, this simple mechanism guarantees that the QoS of the opportunistic Grid is impacted only marginally and in the cases where it improves, it does so by a factor that corresponds to the same improvement measured for the scenario in which it unilaterally exploited the service Grid.

On the other hand, the QoS improvement of the service Grid is maintained for the cases when its workload is in low contention and is, surprisingly, further increased for the high contention workloads. There are two reasons for this result. Firstly, since the jobs of the opportunistic Grid are no longer submitted through the cluster's scheduler, they are now pre-empted by service Grid jobs that are submitted to the clusters. Secondly, under high contention it is likely that all queues of the service Grid cluster will be non-empty. In this situation, BoT jobs from the service Grid workload will be able to use idle nodes of the service Grid (through the opportunistic Grid "interface") and will have their queueing time substantially reduced.

5 Interoperation of gLite and OurGrid Middleware

In this section we present our approach for supporting the interoperation of a service Grid and an opportunistic Grid. We have leveraged on the fact that the NoF is already a part of the OurGrid middleware to implement the simple arbitration mechanism evaluated in Section 4. The main objectives are to allow the usage of the non-dedicated resources provided by an OurGrid-based opportunistic Grid to execute jobs submitted to the gLite-based service Grid and to allow the usage of idle computing cycles from the gLite-based service Grid to execute jobs submitted to the OurGrid-based opportunistic Grid. To achieve these objectives we integrate OurGrid peers into the service Grid and expose the gLite idle resources to the opportunistic Grid. To make the paper self-contained, before we explain the details of our implementation, we give a brief description of the two middleware used.

5.1 Middleware Background

5.1.1 A gLite-Based Service Grid

The gLite middleware was born from the collaborative efforts of more than 80 people in 12 different academic and industrial research centres as part of the EGEE Project [14].

A gLite-based service Grid is composed by a set of resource centres running services to provide remote access to local dedicated computational resources and central services that form the backbone of the service Grid. The gLite Grid services can be thematically grouped into 4 groups: Access and Security Services, Information and Monitoring

Services, Job Management Services and Data Services.

The prime aim of the Access and Security Services is to identify users, allowing or denying access to services, on the basis of agreed policies. It provides credentials having a universal value that works for many purposes across several infrastructures, communities, Virtual Organisations (VOs), and projects. To carry out this task, gLite uses the Public Key Infrastructure (PKI) X.509 technology with Certification Authorities as trusted third parties.

The Information and Monitoring Services provide information about the gLite resources and their status. The published information is used to locate resources and for monitoring and accounting purposes. Much of the data published in the Information Service conforms to a schema that defines a common conceptual data model to be used for resource monitoring and discovery.

The Job Management Services are responsible for dealing with all aspects of the execution of a job on the Grid. The Computing Element (CE) service represents a set of computing resources localised at a resource centre (i.e., a cluster, a computing farm, a SMP machine, etc.) and is responsible for the local job management: (submission, control, etc.). A CE provides a generic interface to the cluster where the cluster itself is represented by a collection of Worker Nodes (WN). Another important service in this group is the Workload Management System (WMS). The WMS is a Resource Broker responsible for the distribution and management of jobs across different resource centres. The purpose of the WMS is to accept user jobs, to schedule them to the most appropriate CE matching user's requirements and Grid nodes availability, to record their status, and to retrieve their output. There are other services in this group used to collect information about the resource usage done by users or groups of users (VOs).

The Data Services manage all aspects related to the location and movement of data among the resource centres of the Grid. A service called Storage Element (SE) is a gLite component that provides a common interface to the storage backend available at the resource centre. It is not uncommon to use an SE to provide remote access to large robots controlling hundreds of terabytes of tape storage. Another data service is the Large File Catalogue (LFC) that keeps meta-data information, mapping a common namespace into the location of each file in the SEs present in the Grid. Last but not least, there is a File Transfer Service (FTS) that is used to establish optimised transfer channels among two SEs in the Grid.

All the gLite Services can be accessed and used through the User Interface (UI) service. The UI provides a set of command-line tools that allow users to authenticate themselves in the Grid, submit jobs and retrieve their output and transfer files to remote Grid resource centres.

The Grid services described above can also be grouped by their scope. Services like Computing Elements and Storage Elements are services with local scope. They must be deployed by each resource centre to provide remote access to their local resources. Service like the Workload Management System, Large File Catalogue and File Transfer Service are global services that are deployed at some core resource centres of the Grid and shared by all users. There are also services like the Access and Security Services and the Information and Monitoring Services that are both local and global services. They need to be deployed at each resource centre and need some global instances to co-ordinate the interaction and propagation of information among local instances.

The gLite job life cycle is tracked by a Job Provenance Service and is described as a sequence of state changes. The Submitted state is attributed to a new job that has just been submitted by the user and is ready to be transferred from the User Interface to the WMS. When the job reaches the WMS its state is changed from Submitted to Waiting, as the job waits to be processed by the WMS. The status is then changed to Ready, indicating that the WMS has chosen a CE to execute the job. The next state is Scheduled, meaning that the job was transferred from the WMS to the CE and is now in the local CE queue, waiting to be scheduled by the Local Resource Management Service (LRMS) within the CE. The job state is changed to Running when it starts to run on the CE resources. Once running the job state can either change to Done or to Aborted. The Done state indicates that the job execution was either concluded, or cancelled by the user. The

Aborted state indicates that the execution was not concluded. Cleared is the final job state, indicating that its output has been collected by the user who submitted the job.

5.1.2 An OurGrid-Based Peer-to-Peer Opportunistic Grid

An opportunistic P2P Grid supported by the OurGrid middleware has three main components, namely the OurGrid Worker, the OurGrid Resource Manager Peer (or simply the OurGrid Peer) and the OurGrid Broker. Figure 1 depicts the OurGrid architecture.

The OurGrid Worker is an agent that executes on the Grid resources (referred as *Grid machines*) and is responsible for implementing a secure environment for the execution of the tasks of Grid applications. A policy defined by the resource owner drives the opportunistic behaviour of the OurGrid Worker. It runs an idleness detection algorithm that triggers the availability of the Grid machine when the resource is idle (as defined by the local policy), and interrupts any Grid task that is executing when, according to the local policy, the resource is deemed not to be idle.

The OurGrid Peer is the component that manages, at each administrative domain (or site), the set of Grid machines that are made available to the Grid by the corresponding site. In general, one peer per site is installed. A peer joins the Grid by notifying a peer discovery service about its existence and it is immediately informed about the presence of other peers on the Grid.

The OurGrid Broker is responsible for providing users with an interface to submit their applications. It is also responsible for performing the scheduling of the applications on the Grid machines and manages the execution of the scheduled applications. A user wishing to run an application must use the OurGrid Broker to connect to a known peer (usually her own site's peer, called the *local* peer), becoming a local user of that peer.

Parallel applications that run on OurGrid are structured as a set of independent tasks (a BoT). There are several ways to interact with the OurGrid Broker to submit applications. However, no matter how the user interacts with the broker, when an application is submitted for execution, the broker contacts its local peer asking for the number of Grid machines required to run the application (this request may carry specific attributes for the machines, such as a particular operating system, or a minimal amount of memory, etc.). When the peer receives such a request it tries to find enough machines that are suitable to execute the application's tasks. If the peer cannot satisfy the request with its own local machines, it tries to

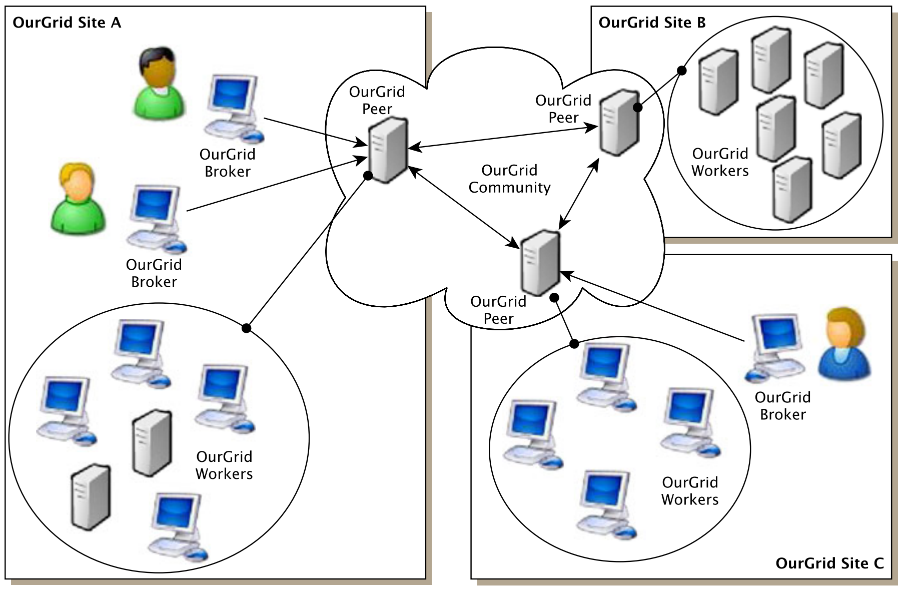

Fig. 1 OurGrid's architecture

obtain machines from other community peers, by forwarding the request to remote peers that may have suitable machines. It then waits for these peers to deliver remote machines. Whenever new machines (local or remote) are made available, the local peer delivers these machines to the requesting broker. As soon as one or more machines are delivered to the broker, it schedules tasks and starts the management of the their execution.

OurGrid has been built to be fast, simple, scalable and secure. It is fast as it allows users to improve the turn-around time of their applications. This capability is provided by features of the OurGrid Broker, such as scheduling with task replication and file transfer optimisations. The system's simplicity is mainly due to its capacity of hiding the Grid heterogeneity behind the OurGrid middleware. In order to achieve scalability, a peer-to-peer approach has been adopted and the Network of Favours, a technology that promotes co-operation among peers, has been designed and implemented. On OurGrid, security is delivered to users by means of the sand-boxing mechanism implemented by the OurGrid Worker. It isolates a potentially malicious application inside a virtual machine, thus protecting the machine and the network from attacks. A more detailed discussion on the OurGrid approach to each one of these issues can be found in [6].

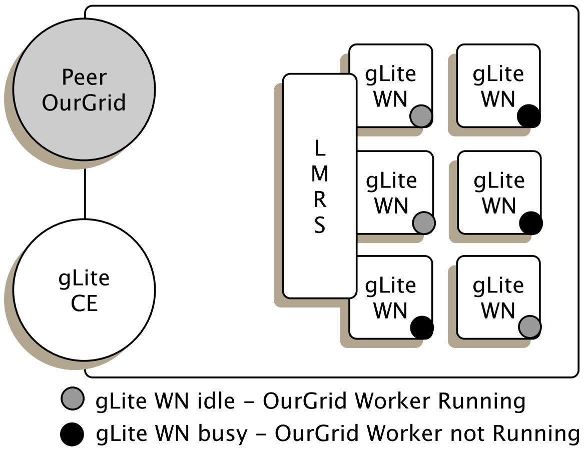
Peer OurGrid
gLite CE
L M R S
gLite WN
gLite WN
gLite WN
gLite WN
gLite WN
gLite WN
gLite WN idle – OurGrid Worker Running
gLite WN busy – OurGrid Worker not Running

Fig. 2 Scavenging idle cycles from the service Grid

5.2 Exposing Dedicated Resources to the Opportunistic Grid

Exposing the service Grid dedicated resources managed by a particular gLite CE to the opportunistic Grid simply requires the installation of an OurGrid Peer and OurGrid Workers in every node managed by the CE, as shown in Fig. 2. In this way the same set of resources would be visible both by the gLite-based service Grid and by the OurGrid-based opportunistic Grid.

However, it is important to define how to prioritise the execution of jobs coming from the service Grid. The OurGrid Workers should be allowed to run in the cluster only in the nodes that are not being used to execute jobs from the service Grid (grey circles in Fig. 2). In gLite-based service Grids the CE is the service responsible for exposing local clusters in the Grid. However, the CE has no power to control how the resources in the cluster are actually used. Instead, the CE delegates this power to the Local Resource Manager System (LRMS) and forwards the remote jobs received trough the Grid to the LRMS queues. So, in order to implement this prioritisation policy we need to modify the way the LRMS manages the cluster resources.

The solution is based on the exploitation of the hooks provided by most LRMSs to execute some commands before and after the execution of the actual job. Prior to the execution of a job in the cluster the LRMS executes a command to stop the OurGrid Worker running in the selected node, then it executes the job on the node and after the execution is concluded and the node is idle again the LRMS starts an OurGrid Worker again in the node. So, the execution of OurGrid Workers does not tag the cluster nodes as busy and whenever a job is scheduled to run in a given node the OurGrid Worker running there is interrupted to be restarted only after the cluster jobs are finished.

With this scheme, whenever a node in the cluster is idle it will be running an OurGrid Worker and will be ready to run jobs coming from the opportunistic Grid. On the other hand, whenever a node in the cluster is needed to execute a job coming from the service Grid it will be readily available as well.

6 Related Work

With the proliferation of Grid middleware distributions, interoperability between different Grid infrastructures has become a major concern. To create mechanisms to allow a seamless interoperation between those infrastructures is the next step needed to provide the ubiquitous access to computational resources and services pledged by the Grid computing model.

Several interoperability efforts are actually in course to allow specific Grid infrastructures to interoperate using different strategies. The first one is to try to find the least common denominators between the Grid services provided by the targeted infrastructures and develop new services to connect them at different levels. This is the case of the work carried out by the Grid Interoperability Now (GIN) working group. The GIN group has being working to provide interoperability between 9 service Grid infrastructures spread all over the globe [13]. At the time of writing GIN has achieved interoperability of the Grid information systems using the gLite BDII service as a common information provider. The current work involves the connection of the infrastructures allowing common job submission, data movement and authorisation and identity management.

Another example of this approach is presented by Riedel et al. [22]. Their work describes the interoperability between the gLite-based EGEE infrastructure and the UNICORE-based Distributed European Infrastructure for Supercomputing Applications (DEISA). DEISA is a computing infrastructure formed by the assembling of large dedicated computing centres, clusters and super-computers mainly focused on the execution of tightly-coupled jobs. This activity is being conducted with the objective of combining the power of the two infrastructures to run biology experiments for developing new drugs to combat Malaria. Marzolla et al. [21] provides more details on how the interoperability between gLite and UNICORE is being conducted.

The Open Middleware Infrastructure Institute for Europe (OMII-Europe)[8] was a project dedicated to the development of components to enable the interoperability between three Grid middleware (gLite, Globus and UNICORE) used by several Grid infrastructures in Europe [12].

The second approach found in the literature describes the development of meta-schedulers able to submit jobs to different Grid infrastructure at the same time, connecting them at the level of the user interface. GridWay [26], GrADS [25], P-GRADE [17], and GANGA [5] are just some examples of meta-schedulers for Grid infrastructures.

All the related works presented so far were targeted to combine the resources provided by different service-Grid infrastructure. The work conducted in the context of the EDGeS Project[9] goes in another direction, closer to our approach, as it also targets the combination of non-dedicated resources and service Grids. The main difference between EDGeS and our approach is that EDGeS targets the utilisation of resources donated through voluntary computing, where users have no incentive to donate their resources except for helping in the execution of a specific, usually big and well known, application [11, 24]. Our approach targets users who donated their idle resources with the aim to gain access to the idle resources donated by others when they need to run their applications using the Grid. Nevertheless, the approach promoted by the EDGeS project can also be used to integrate P2P Grids and service Grids [8].

Finally, unlike all previous works on interoperability of Grid systems that study how Grids should interoperate, the focus of our work is mainly on how to enable this interoperation in a fair way.

7 Concluding Remarks

In this paper we presented an approach for supporting the interoperation of a service Grid and an opportunistic Grid. We propose a simple arbitration mechanism that has several advantages. Firstly, it allows idle resources belonging to the

[8]http://omii-europe.com/.

[9]http://www.edges-grid.eu/.

service Grid to be used in an opportunistic way; secondly, the provision of an opportunistic Grid allows shared resources to be added to the infrastructure, a feature that turns out to be very important for consortia in which many of the member institutions can not afford the provision of dedicated resources. Finally, as our simulation results indicate, it is possible to use the opportunistic Grid to drain BoT jobs out of the service Grid queues, greatly reducing the execution time of this jobs.

The proposed approach has been implemented in the context of the EELA-2 project (http://www.eu-eela.eu/), an initiative co-funded by the European Commission within its Seventh Framework Programme and involving 78 institutions from 11 countries in Latin America and 5 in Europe.

Acknowledgements Authors would like to thank the European Commission for the co-funding of both the EELA and the EELA-2 projects. Special thanks to the NorduGrid team that made available the traces of their system in the Grid Workloads Archive. Francisco Brasileiro would like to thank the financial support from CNPq/Brazil (grant 300.646/96).

References

1. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@home: an experiment in public-resource computing. Commun. ACM, **45**(11), 56–61 (2002)
2. Anderson, D.P.: BOINC: a system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID '04, pp. 4–10. IEEE Computer Society, Washington, DC, USA (2004)
3. Andrade, N., Brasileiro, F., Cirne, W., Mowbray, M.: Automatic Grid assembly by promoting collaboration in peer-to-peer Grids. J. Parallel Distrib. Comput. **67**(8), 957–966 (2007)
4. Anglano, C., Canonico, M., Guazzone, M.: The ShareGrid peer-to-peer desktop Grid: infrastructure, applications, and performance evaluation. J. Grid Computing **8**(4), 543–570 (2010). doi:10.1007/s10723-010-9162-z
5. Brochu, F., Egede, U., Elmsheuser, J., Harrison, K., Jones, R.W.L., Lee, H.C., Liko, D., Maier, A., Moscicki, J.T., Muraru, A., Patrick, G.N., Pajchel, K., Reece, W., Samset, B.H., Slater, M.W., Soroko, A., Tan, C.L., Vanderster, D.C.: Ganga: a tool for computational-task management and easy access to Grid resources. CoRR. abs/0902.2685 (2009)
6. Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., Mowbray, M.: Labs of the world, unite!!! J. Grid Computing **4**(3), 225–246 (2006)
7. da Silva, D.P, Cirne, W, Brasileiro, F.: Trading cycles for information: using replication to schedule bag-of-tasks applications on computational Grids. In: Proceedings of the Euro-Par 2003: International Conference on Parallel and Distributed Computing, pp. 169–180. Klagenfurt, Austria (2003)
8. de Barros, A.G., Furtado, A.A., Brasileiro, F.: Bridging OurGrid-based and gLite-based Grid infrastructures. In: Proceedings of the Second EELA-2 Conference (2009)
9. Epema, D.H.J., Livny, M., van Dantzig, R., Evers, X., Pruyne, J.: A worldwide flock of condors: load sharing among workstation clusters. Future Gener. Comput. Syst. **12**(1), 53–65 (1996)
10. Estrada, T., Taufer, M., Anderson, D.: Performance prediction and analysis of BOINC projects: an empirical study with EmBOINC. J. Grid Computing **7**, 537–554 (2009). doi:10.1007/s10723-009-9126-3
11. Farkas, Z., Kacsuk, P., Balaton, Z., Gombás, G.: Interoperability of BOINC and EGEE. Future Gener. Comput. Syst. **26(**8), 1092–1103 (2010)
12. Field, L., Laure, E., Schulz, M.: Grid deployment experiences: Grid interoperation. J. Grid Computing **7**, 287–296 (2009). doi:10.1007/s10723-009-9128-1
13. Flechl, M., Field, L.: Grid interoperability: joining Grid information systems. J. Phys. Conf. Ser. **119**(6), 062030 (2008)
14. Gagliardi, F., Jones, B., Grey, F., Bégin, M.-E., Heikkurinen, M.: Building an infrastructure for scientific Grid computing: status and goals of the egee project. Philos. Trans. R. Soc. Lond. A: Math. Phys. Eng. Sci. **363**(1833), 1729–1742 (2005)
15. Hernández, V., Blanquer, I., Aparicio, G., Isea, R., Chaves, J.L., Hernández, Á., Mora, H.R., Fernández, M., Acero, A., Montes, E., Mayo, R.: Advances in the biomedical applications of the EELA project. Stud. Health Technol. Inform. **126**, 31–36 (2007)
16. Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., Epema, D.H.J.: The Grid workloads archive. Future Gener. Comput. Syst. **24**(7), 672–686 (2008)
17. Kacsuk, P., Kiss, T., Sipos, G.: Solving the Grid interoperability problem by P-GRADE portal at workflow level. Future Gener. Comput. Syst. **24**(7), 744–751 (2008)
18. Litzkow, M., Livny, M., Mutka, M.: Condor—a hunter of idle workstations. In: Proceedings of the 8th International Conference of Distributed Computing Systems, pp. 104–111. IEEE Computer Society, San Jose, CA, USA (1988)
19. Marechal, B., Bello, P.H.R., Carvalho, D.: Building a Grid in latin america: the EELA project e-infrastructure. In: CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, pp. 835–839. IEEE Computer Society, Washington, DC, USA (2007)

20. Marechal, B., Bello, P.R., Carvalho, D., Mayo, R.: Applications ported to the EELA e-infrastructure. In: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07), pp. 852–857. IEEE Computer Society, Washington, DC, USA (2007)
21. Marzolla, M., Andreetto, P., Venturi, V., Ferraro, A., Memon, A.S., Memon, M.S., Twedell, B., Riedel, M., Mallmann, D., Streit, A., van de Berghe, S., Li, V., Snelling, D., Stamou, K., Shah, Z.A., Hedman, F.: Open standards-based interoperability of job submission and management interfaces across the Grid middleware platforms gLite and UNICORE. In: Proceedings of International Interoperability and Interoperation Workshop (IGIIW) 2007 at 3rd IEEE International Conference on e-Science and Grid Computing, pp. 592–599. IEEE Computer Society, Bangalore, India (2007)
22. Riedel, M., Memon, A.S., Memon, M.S., Mallmann, D., Streit, A., Wolf, F., Lippert, Th., Venturi, V., Andreetto, P., Marzolla, M., Ferraro, A., Ghiselli, A., Hedman, F., Shah, Z.A., Salzemann, J., Da Costa, A., Breton, V., Kasam, V., Hofmann-Apitius, M., Snelling, D., van de Berghe, S., Li, V., Brewer, S., Dunlop, A., De Silva, N.: Improving e-science with interoperability of the e-infrastructures EGEE and DEISA. In: Proceedings of the 31st International Convention MIPRO, Conference on Grid and Visualization Systems (GVS), pp. 225–231. Opatija, Croatia, Croatian Society for Information and Communication Technology, Electronics and Microelectronics (2008)
23. Santos, R., Andrade, A., Cirne, W., Brasileiro, F., Andrade, N.: Relative autonomous accounting for peer-to-peer Grids: research articles. Concurr. Comput.: Pract. Exp. **19**, 1937–1954 (2007)
24. Urbah, E., Kacsuk, P., Farkas, Z., Fedak, G., Kecskemeti, G., Lodygensky, O., Marosi, A., Balaton, Z., Caillat, G., Gombas, G., Kornafeld, A., Kovacs, J., He, H., Lovas, R.: EDGeS: bridging EGEE to BOINC and XtremWeb. J. Grid Computing **7**, 335–354 (2009). doi:10.1007/s10723-009-9137-0
25. Vadhiyar, S.S., Dongarra, J.J.: A metascheduler for the Grid. In: HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, p. 343. IEEE Computer Society, Washington, DC, USA (2002)
26. Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated harnessing of the IRISgrid and EGEE testbeds with gridway. J. Parallel Distrib. Comput. **66**(5), 763–771 (2006)
27. Wang, Y., Scardaci, D., Yan, B., Huang, Y.: Interconnect EGEE and CNGRID e-infrastructures through interoperability between gLite and GOS middlewares. In: Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing (e-Science'07)—International Grid Interoperability and Interoperation Workshop. IEEE Computer Society, Los Alamitos, CA, USA (2007)