A Decentralized Model for Scheduling Independent Tasks in Federated Grids ⋆

Katia Leal [a,*], Eduardo Huedo [b], Ignacio M. Llorente [b]

[a] *Universidad Rey Juan Carlos, Departamento de Sistemas Telemáticos y Computación, Escuela Superior de Ciencias Experimentales y Tecnología, Tulipán SN, 28933 Mósteles, Madrid, Spain*

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

Abstract

In this paper we present a decentralized model for scheduling independent tasks in Federated Grids. This model consists in a set of meta-schedulers on each of the grid infrastructures of the Federated Grid. Each meta-scheduler has to implement a mapping strategy in order to improve two of the most common objective functions of tasks scheduling problems: makespan and resource performance. We consider four possible algorithms that have to provide a simple, decoupled, and coarse-grained solution that could be deployed in any Grid. The main axis of the algorithms is that they consider the *performance* of the infrastructures forming the Federated Grid, not only their *state*.

Key words: Federated Grids, Dynamic Objective, Advance Scheduling, Independent Tasks, Decentralized Model

⋆ This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia and through the research grant TIN2006-02806. Research work also supported by the Madrid Regional Research Council under project TIC-000285-0505.

* Corresponding author.
Email address: katia.leal@gmail.com (Katia Leal).

1 Introduction

Recent trends in planning and scheduling research reflect movements from isolated multi-host scenarios to open large scale infrastructures with several computational resources. These approaches have brought about a change from local to global scheduling. Although the scheduling problem has been around for some time [46], we can only utilize some of the ideas found in the literature. The principle reason why we cannot take advantage of earlier research is because assumptions underlying centralized systems do not hold in Grid circumstances and produce poor Grid schedules in practice [3], and this is the reason why several models have been proposed to meet the requirements of these new scenarios [12,2].

A system can be considered a Grid only if it coordinates resources that are not subject to centralized control, by using standard, open, general-purpose protocols and interfaces, and to deliver nontrivial qualities of service [17]. Thus, a Computational Grid provides a common layer to integrate heterogeneous platforms through the definition of a consistent set of abstraction and interfaces for access and management of shared resources. This layer is not limited to a specific administrative domain - it may be possibly distributed across several domains in all platforms.

The evolution of Grid Computing can be classified in three stages [37]:

- ❒ *Enterprise Grid*: the aggregation of various local grids allows inter-department resource sharing within the same organization. Internal resources are managed by different local resource management systems that could be geographically distributed.
- ❒ *Partner Grid*: the collaboration among several organizations, universities, and research centers whose objective is to provide resource sharing. The distributed resources belonging to several organizations are managed by different local resource management systems. EGEE at the European Union [13], TeraGrid at the United States [45], e-Science at the United Kingdom [15], DAS-2 at Netherlands [9], D-Grid at Germany [39] and GridX1 at Canada [1], are some examples of Partner Grids.
- ❒ *Utility Grid*: external service providers allow access to superior computing performance to satisfy peak or unusual demands.

The *Federated Grid* represents the next stage in the evolution of Grid Computing. It is characterized by allowing resource sharing among several grid infrastructures of different types. Thus, a Federated Grid supports the interconnection of Enterprise, Partner and Utility Grids to increase the total number of participating resources. Otherwise, this powerful resource could not be obtained by the separate grid resources. Finally, the same requirements for

Grid systems should apply for Federated Grids [17]. Federated Grids are fully reviewed in Section 2, where we identify and explain all of the elements of a Federated Grid.

Section 4 investigates the scheduling problem in Federated Grids. We propose a generic *decentralized* model that is different than the *centralized*, *application-centric* and *ad-hoc* solutions. These earlier solutions are discussed in Section 3. Our approach situates a meta-scheduler on the top level of the system architecture. In contrast to local schedulers and workload managers, which possess complete knowledge of system state and user requests, our meta-scheduler will have general information about the entire Federated Grid. This is why we cannot try to apply fine-grained techniques that are more suitable to local schedulers or workload managers that completely control the resources. Instead, this obstacle is overcome with light, decoupled, and coarse-grained techniques. Various algorithms that follow this idea and that could be hosted in this meta-scheduler are presented in Section 5. The four proposed algorithms are mainly based on a performance model [35] that allows to parametrize and compare the different grids forming a Federated Grid. In this way, we characterize the performance of a Federated Grid by means of the r_∞, and $r_{1/2}$ parameters [24] in order to determine the number of jobs, the *objective* to submit to each of the grid infrastructures forming the Federated Grid.

Next, we enumerate the four proposed algorithms. Since these algorithms are generic, they could be hosted in any meta-scheduler:

- ❒ The **Static Objective (SO)** algorithm works out the objective off-line. Simulation results showed that this *static* scheduling policy maximized the throughput of internal resources, without decreasing the computational time, and provided a fair distribution of the jobs [31], compared with GridWay's current scheduling policy.
- ❒ The **Dynamic Objective (DO)** algorithm calculates the objective in execution time. Thus, as we will see in Section 7, it solves some problems that were encountered in the static version. Even though it provides a fair distribution of the jobs that maximized the throughput of internal resources, it does not reduce the makespan of the applications.
- ❒ The **Static Objective and Advance Scheduling (SO-AS)** algorithm combines the SO algorithm with an Advance Scheduling (AS) technique to reduce the makespan achieved in the static version. In this case, the algorithm does not wait for a free node, instead, it queues jobs in advance to avoid the latencies that are inherent to Federated Grids. Later on we will show that this version only reduces the makespan in the case of high saturation in the external resources.
- ❒ The **Dynamic Objective and Advance Scheduling (DO-AS)** version combines the two mechanisms that have demonstrated to be an enhancement: DO and AS. Thus, DO-AS compared with GridWay's current

scheduling policy, clearly satisfies the makespan and performance objectives of the resources [30], which are two of the most common objective functions of tasks scheduling problems [12].

A full description of these algorithms with detailed explanations are provided in Section 5. All of the algorithms follow these rules:

- ❒ *Do not require configuration information*: users, nor programmers have to provide configuration information, for example, the tasks length. The scheduler does not need node specific information, such as processor speed.
- ❒ *Have to be simple*: the pseudo code of the algorithms shows simple calculation operations that can be easily implemented in a few code lines.
- ❒ *Have to adapt* to grid resources performance.

In Section 6, we explain how we deployed our decentralized model of meta-schedulers that implement the proposed algorithms, to schedule independent tasks on a simulated Federated Grid by means of the GridSim toolkit [21].

Section 7 provides the simulation results of all the algorithms and determines the best option. The four algorithms are compared amongst themselves and with GridWay's current scheduling policy. However our comparison does not end with the makespan achieved: we also calculate the objective for each algorithm and show graphically the difference in their behavior.

Section 8 summarize the benefits of our decentralized model in conjunction with the best scheduling algorithm. Guidelines for our current research activities are also provided.

2 Federated Grids

In this Section we will identify the different participating elements of a Federated Grid and explain their roles. Figure 1 depicts a standard Federated Grid consisting of only two Grid infrastructures, which is sufficient for our analysis of the components of a Federated Grid. Enterprise, Partner or a Utility Grids can be used as the two Grid infrastructures making up the Federated Grid.

The following acronyms are used in Figure 1 and we now provide their definitions:

- ❒ *LRMS*: Local Resource Management System, scheduler, local scheduler, local resource manager, local workload manager. These tools are usually cluster managers that were taken from high-performance and distributed computing, and now are generally used in Grid Systems. The

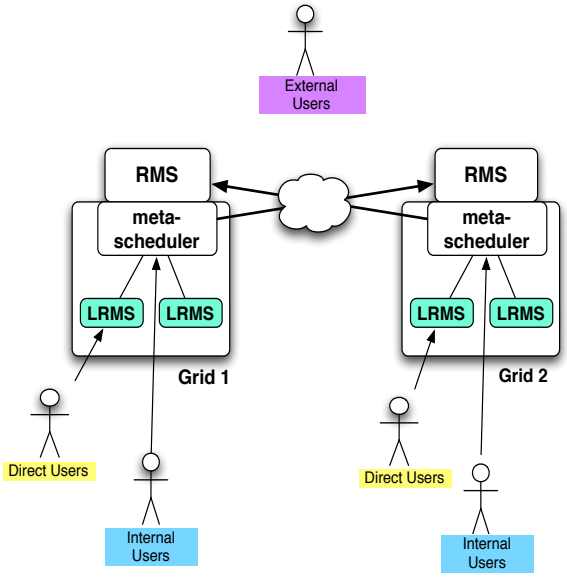

Fig. 1. Example of a Federated Grid.

most widespread include: Sun's Sun Grid Engine (SGE) [44], Platform's Load Sharing Facility (LSF) [34] and Veridian's Portable Batch System (PBS) [36].

- ❒ *meta-scheduler*: one or more components of a grid middleware. It can also be an external tool that uses middleware components or services.
- ❒ *RMS*: Resource Management Service. This is usually a component of a grid middleware that provides an interface for requesting and using remote system resources for the execution of jobs. An example is the *Grid Resource Allocation and Management* (GRAM) service [41] in the Globus Toolkit [22].

The meta-scheduler perspective allows two types of resources to be identified in Figure 1:

- ❒ *Internal Resources*: these are the resources directly accessible by the meta-scheduler through the corresponding LRMS. That is, the resources owned by the particular research center, laboratory or company.
- ❒ *External Resources*: those resources not directly accessible by the meta-scheduler. That is, the resources belonging to other organizations, universities or research centers.

The meta-scheduler in Grid 1 sees internal resources as the foundation for Grid 1. These resources are directly accessible via the corresponding LRMS, such as SGE or PBS, possibly through a uniform interface. External resources are all those resources accessible through the RMS specification. In this case, the meta-scheduler at Grid 1 only perceives one external resource, Grid 2.

The users depicted in Figure 1 can be classified as internal, external, and direct. Differences are found in the way they access resources and their rights to these resources:

- ❒ *Internal Users*: are those users that submit jobs directly to the meta-scheduler. The jobs can then be executed in both internal and external resources.
- ❒ *External Users*: all those users that submit jobs through the RMS interface to external resources. Actually, the meta-scheduler is the one that decides to submit jobs to external resources: this entire process is completely transparent for internal users. This is how internal users became external users.
- ❒ *Direct Users*: are all those users that submit jobs directly to the final resources through the LRMS, bringing about changes in influence in the load of the resources.

3 Related Work

In this Section we will review centralized, application-centric and some ad-hoc attempts which function among operational grids. Some approaches to enable grid interoperability are also introduced.

3.1 Centralized approaches

Centralized approaches are fine-grained solutions more suitable for local schedulers or workload managers than for scheduling in Federated Grids. This is because they present scalability problems.

One of the factors which determine whether an approach is worth considering is the time required to evaluate a solution [8,7]. Genetic Algorithm (GA) is one of the most popular mechanisms used for scheduling independent jobs in a grid environment because of its simplicity, however its time-consuming iteration should not be ignored. In fact, the Improved Genetic Algorithm (IGA) was proposed to enhance the search performance of conventional GA [51]. However, an experiment using 8 resources and 60 tasks reveals a worst time near to 1 second. Other results with additional resources and tasks showed that more time is required to evaluate a solution. Later sections will show that the performance of DO-AS is independent of the number of jobs, and resources: the calculation of a new objective is an extremely light process. Thus, unlike GA, DO-AS is fast enough to be used in a realistic scheduling with hundreds of jobs and resources.

Since Min-min, and Max-min [19] are static algorithms, the assignment of tasks is fixed a priori: these algorithms need prediction information on processor speeds and task lengths to compute tasks priorities. However, a cost estimate based on static information is not adaptive to situations such as one of the nodes selected to perform a computation fails and becomes isolated from the system due to network failure, or is so heavily loaded with jobs that its response time becomes longer than expected. Another approach that does not require such information was proposed to improve the previous ones, but the simulation results demonstrated that it was only the second best algorithm [20]. In contrast, DO-AS is dynamic, does not require information on processor speeds and task lengths, and the results we show have been obtained from a simulated Federated Grid based on a real testbed.

Several models have also been proposed to meet the requirements of the new Grid scenarios, as mentioned in [23]. Some of the most frequently cited models include:

- *Global backfilling*: as depicted in Figure 2(a), users submit jobs to a specific host with a given policy. A controller may also be backfilling jobs among the different resources [52]. Backfilling strategies typically require a job runtime estimation which is provided by the user. In contrast, our approach does not require any estimation information: it is completely transparent for users, and applications.
- *Global scheduler*: as seen in Figure 2(b) this is the classical brokering approach in which users submit jobs to a global scheduler that will later submit the job to the corresponding local resource. Thus, jobs are queued at two different levels: first at the global scheduler, and then at the local scheduler queue. Two different resource selection algorithms are proposed [42]: first jobs are processed in order of arrival to the meta-scheduler, then when a job arrives at the second level, it is submitted to N different sites. The start of the job in one site results in the cancellation of the remaining submissions. This implies lot of communications per job submitted to the second level. We propose a decoupled solution on which schedulers at different levels are independent: local schedulers, as well as workload managers do not need to modify their policies. Thus, we can deploy our approach in any Grid.
- *Global dispatcher*: in this schema all jobs are submitted to a centralized dispatcher, and no local schedulers are instantiated, as depicted in Figure 3(c). The goal of the global dispatcher is to find the allocation that minimizes the job start time, but using a unique centralized reservation table: local centers queues are of size zero [14]. Other approaches [43] use a central dispatcher to analyze different tasks mapping policies.
- *Global queue + pull mechanism*: as seen in Figure 3(d), this scenario also reveals only a unique global queue, but in this case local computational schedulers pull jobs from it when there are enough available resources to

run the jobs. Users can submit jobs to a global queue from which local schedulers dynamically pull jobs on demand [38]. This schema imposes the use of a particular policy at the local scheduler level, thus it is not compatible with other legacy local schedulers.

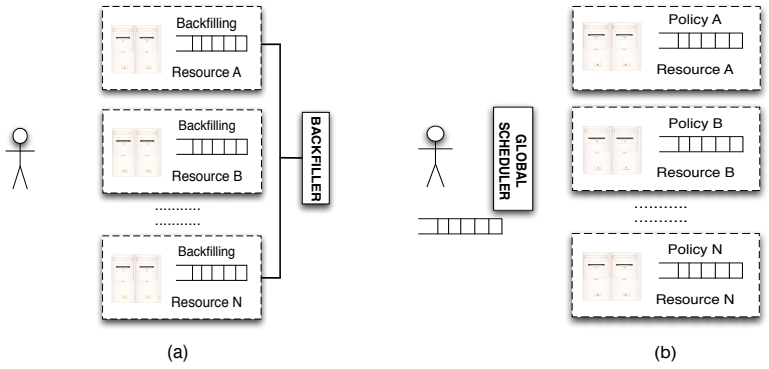

Fig. 2. (a) Global Backfilling, (b) Global Scheduler.

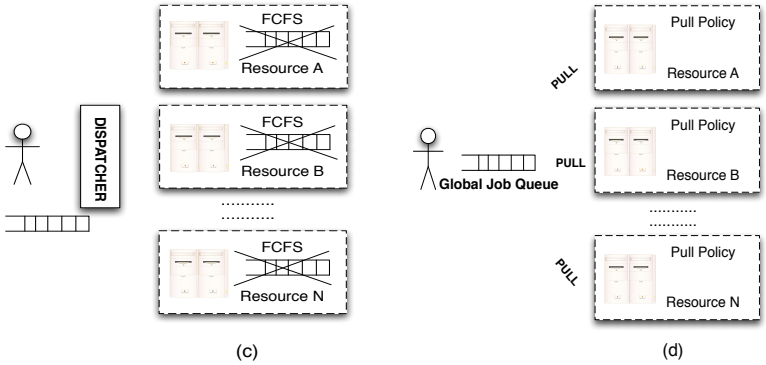

Fig. 3. (c) Global Dispatcher, (d) Global queue + pull mechanism.

These four approaches typically impose a particular configuration at the local level, while our solution is compatible with legacy local schedulers and workload managers.

3.2 Application-centric approaches

Scheduling strategies are not limited to *centralized* solutions. Other models propose *application-centric* [4], or *job-guided* strategies [23]. The goal of application-centric solutions is to promote the performance of an individual application rather than to optimize the use of system resources or to enhance the performance of a stream of jobs. Instead of using a standard framework, such as the Distributed Resource Management Application API (DRMAA) [40], these applications have to be developed by using a particular framework.

In the job-guided approach each job has its own dispatcher that is aware of the

states of the different resources, but is unaware of any other jobs that have been submitted. On the other hand, the accuracy of our approach depends on the information provided by already finished jobs and those that have to be scheduled. The job-guided approach also requires interaction between the job dispatcher and the local managers, imposing the implementation of a particular strategy at the local level to obtain better results. We do not impose a two level scheduling model since several Grid infrastructures forming the Federated Grid can use their own legacy local or meta-scheduler policies at the different layers.

3.3 Grid Interoperability solutions

Earlier successful efforts at exploiting federated grids have been documented in the literature [6,5,18]. However, the ability to federate in these cases is the result of an effort focused on adapting applications to the distinct individual grids at the user and middleware level. As a consequence, all these *ad-hoc* approaches to interoperate are not scalable: the probability of success is likely to decrease with every additional independent grid. The scheduling strategies are also not clear, since applications were modified to utilize a specific grid configuration. In fact, an argument could be made that it is not even a Federated Grid since the interoperability is not scalable, but requires manual effort. In contrast, we propose a scheduling approach that enables Federated Grids without changing the programming model nor the application code.

InterGrid [10] promotes interlinking of Grid systems through *InterGrid Gateways* that mediate resource access based on peering arrangements between different Grids. InterGrid philosophy advocates a decentralized resource management: a Grid requires a means to specify its policies, defining which resources are available to other Grids under which circumstances, and to make them available to other Grids, but retaining ultimate control over its resources and to whom it wants to provide access. However, this approach reflects a more economical view, where business application support is a primary goal. In contrast, with our policies we want to improve the makespan of the applications and the performance of the resources.

The Meta-Brokering approach [28] supports grid interoperability by means of a higher level brokering service, the *Meta-Broker*. The Meta-Broker, inter-grid broker or inter-grid gateway, sits on top of the resource brokers and uses meta-data from them to decide where to send a user job. Its scheduling philosophy is called meta-brokering, and creates a meta-level above current resource management solutions by using technologies from the area of the semantic web. This approach provides data about resource managers in the form of meta-data. This is a centralized solution that does not stand Grid systems

requirements.

4 Decentralized Model

The previously reviewed approaches do not present suitable architecture for scheduling in Federated Grids. We propose the *decentralized* model depicted in Figure 4 as an alternative to centralized, application-centric or ad-hoc solutions to this problem. The model puts a meta-scheduler at the top level of each grid infrastructure, over the workload managers. This new layer presents a queue on which jobs have to wait to be scheduled. However, the experimental results will demonstrate that, even with the delays introduced by the different queues jobs have to pass through, this model is still very efficient. Instead of having a unique centralized global scheduler to map the jobs of the distinct Grids, each one has its own meta-sceduler. The aim of the mapping strategy implemented on the meta-scheduler at each Grid infrastructure of the Federated Grid is to reduce the makespan of its applications and to increase the performance of its own Grid infrastructure.

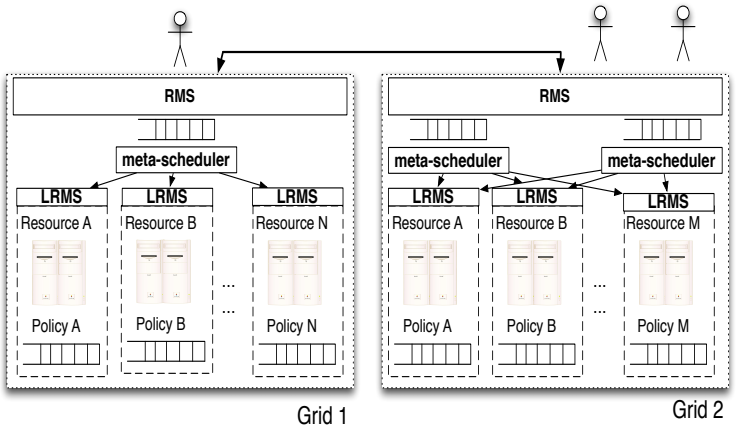

Fig. 4. Decentralized model.

To summarize, we proposed a decentralized approach like the one depicted in Figure 4 on which:

❶ A meta-scheduler on each grid infrastructure of the Federated Grid implements the appropriate scheduling algorithm, as mentioned before.
❷ For users and applications the mechanism for mapping jobs on the Federated Grid is completely transparent.
❸ Each algorithm implemented on the meta-scheduler at each Grid infrastructure of the Federated Grid is aware of reducing the makespan of its own applications, and of increasing the performance of its own Grid infrastructure.

❹ The mapping strategy has to be simple, has to adapt to grid resources performance, and does not require configuration information.

Even though we offer results based on a simulated testbed, the key contribution is to implement our mapping strategies in a real meta-scheduler. Thus, from the very beginning we decided to use *GridWay* [25] as the meta-scheduler of our decentralized model. Among other possible meta-schedulers, we chose GridWay because it presents an independent scheduling module that facilitates the implementation and incorporation of new scheduling policies. GridWay also performs job execution management and resource brokering on a grid consisting of distinct computing platforms managed by Globus [22] services. GridWay allows unattended execution of single, array, or complex jobs on heterogeneous and dynamic grids. A Web Service - Grid Resource Allocation and Management (WS-GRAM) Globus Toolkit service hosting a GridWay meta-scheduler acts as a grid gateway between two grid infrastructures [33,26]. We call this entity the *GridGateWay*. GridGateWay acts as a gateway between the different grid infrastructures forming the Federated Grid, enabling submission, monitoring and control of jobs across the grids that make up the Federated Grid. GridGateWay will be managed as a common resource in the first grid infrastructure, and will act as a proxy of its users in the second grid. The second grid infrastructure will appear in the information system of the first as a normal WS-GRAM publishing an aggregation of grid resources. Thus, we can implement the Federated Grid depicted in Figure 4 by using GridWay as the meta-scheduler, and the WS-GRAM as the RMS interface. Other meta-schedulers [2,27] could be used in place of GridWay. Other remote submission interfaces, such as OGF Basic Execution Service (BES) [16], could be used instead of GRAM.

5 Algorithms for scheduling on Federated Grids

We have applied a performance model [35] to GridWay's current scheduling policy to obtain four different algorithms. The performance model allows to parametrize and compare the different Grids forming a Federated Grid. One of the resultant algorithms satisfactorily achieves two of the most common objective functions of tasks scheduling problems [12]: makespan and performance of the resources.

5.1 GridWay's current scheduling policy

Normal is the algorithm that implements GridWay's current mapping strategy. This algorithm maps no more than `DISPATCH_CHUNK` jobs to resources

every SCHEDULING_INTERVAL seconds, if there are unscheduled jobs. The algorithm maps the next job first to internal resources. Thus, if there are available free nodes in internal resources, the job is mapped to one of them. Otherwise, maps the job to the next free external resource. Consequently, this algorithm does not take into account the past performance of the different participating infrastructures, instead it only determines if there are free nodes or not. Once the job has been submitted to the corresponding free node of an internal or external resource, a mechanism of *migration* controls that the job starts its execution before the elapse of SUSPENSION_TIMEOUT seconds. If this time is exceeded, the job is cancelled and moved to the unscheduled jobs list to be scheduled again.

This algorithm was not designed for scheduling in Federated Grids, and does not provide good results under this scenario. However, it can work well in a Grid if it has direct access to resources, and these can be used in an opportunistic manner.

5.2 Static Objective: SO

Objective is a variable that determines the number of jobs that should be submitted to each of the grid infrastructures forming the Federated Grid. Objective maximizes the throughput of the internal resources without increasing the makespan. We used the equation that represents the best characterization of the Federated Grid to obtain these numbers. The characterization can be obtained if we take the line that represents the average behavior of the system, as proposed by Hockney, and Jesshope [24]:

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

In Equation 1 n represents the number of completed tasks as a function of time t. The other parameters are:

- **Asymptotic performance** r_∞: is the maximum rate of performance in tasks 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** $n_{1/2}$: is the number of tasks required to obtain the one-half of the asymptotic performance. This parameter is also a measure of the amount of parallelism in the system as seen by the application.

The linear relation represented by Equation 1 can be used to define the performance of each grid infrastructure forming the Federated Grid (tasks completed per second), the *aggregation* or *federation* model [48] is then used to determine

the Objective even in case of having more than two participants.

Since this algorithm is static, we need to train the testbed in order to obtain the performance of the different infrastructures forming the Federated Grid. We first run the Normal algorithm to obtain the linear equations of each participant of the Federated Grid. Then, by means of the aggregation model we work out off-line the objective for the proposed application. That is, we determine the number of jobs of that application that should be submitted to each participant of the Federated Grid, in order to increase the throughput of internal resources without increasing the makespan.

Once we have calculated off-line the objective, we update the SO algorithm with this information and execute it. At runtime, SO maps no more than `DISPATCH_ CHUNK` jobs to resources every `SCHEDULING_INTERVAL` seconds if there are still unscheduled jobs. In a Federated Grid, GridWay can receive jobs from internal and external users. If the job to schedule is internal, there are available internal resources, and it have not been met the internal objective, then job is scheduled to an internal resource. Otherwise, if there are no available internal resources or the internal schedule objective has been met, SO checks the possibility of submitting the job to an external resource. Thus, if there are available external resources, and the external objective has not been met, the job is scheduled to an external resource. Otherwise, if initially the job was not internal, and in order to avoid the situations on which a participant of the Federated Grid can receive from another one a job previously submitted, GridWay only schedules external jobs to internal resources. Finally, all the jobs that are being scheduled are dispatched, and SO programs a new event for the next schedule.

5.3 Dynamic Objective: DO

DO calculates the objective dynamically at execution time, instead of off-line as is the case with the SO algorithm.

Pseudo code for the DO algorithm is given in Algorithm 1, which shows how to calculate a new objective every `OBJECTIVE_INTERVAL` seconds if there are jobs to schedule (line 1), and if there are enough samples from jobs already finished (line 3) in each grid infrastructure to properly characterize the corresponding grid. The algorithm first calculates the linear relation of Equation 1 for internal resources to define its performance (line 4-7). Lines 8 to 11 do the same for the external resources. Then, in lines 12 to 13 the algorithm calculates the linear equation of the whole Federated Grid as an *aggregation* from the linear relations of each infrastructure forming the Federated Grid. Any problem calculating the linear relations results in using a default objective. Once the

Algorithm 1: UpdateObjective()

```
1   if (unscheduledJobs.size() < 1) then
2       sendInternalEvent(OBJECTIVE_INTERVAL, UPDATE_OBJECTIVE);
3   if ((internalX.size() >= MIN_IN_SAMPLES) && (externalX.size() >= MIN_EX_SAMPLES)) then
4       internalRegression = linearEquation(internalX, internalY);
5       if (internalRegression.getR_Inf() < 0 || internalRegression.getN_1_2() > 0) then
6           defaultPrediction();
7           sendInternalEvent(OBJECTIVE_INTERVAL, UPDATE _OBJECTIVE);
8       externalRegression = linearEquation(externalX, externalY);
9       if (externalRegression.getR_Inf() < 0 || externalRegression.getN_1_2() > 0) then
10          defaultPrediction();
11          sendInternalEvent(OBJECTIVE_INTERVAL, UPDATE _OBJECTIVE);
12      r_inf_FG = internalRegression.getR_Inf() + externalRegression.getR_Inf();
13      n_1_2_FG = internalRegression.absN_1_2() + externalRegression.absN_1_2();
14      X = (unscheduledJobs.size() + n_1_2_FG) / r_inf_FG;
15      newInternalObjective = (internalRegression.getR_Inf() * X) + internalRegression.getN_1_2()
16      if (newInternalObjective > unscheduledJobs.size()) then
17          newInternalObjective = unscheduledJobs.size();
18      newExternalObjective = unscheduledJobs.size() - newInternalObjective;
19      if (newExternalObjective < 0) then
20          newExternalObjective = 0;
21      internalObjective = newInternalObjective;
22      externalObjective = newExternalObjective;
23  sendInternalEvent(OBJECTIVE_INTERVAL, UPDATE_OBJECTIVE);
```

performance of the Federated Grid has been defined, the algorithm can determine the time for the Federated Grid to complete `unscheduledJobs.size()` jobs (line 14). In lines 15 to 17 the algorithm determines the jobs that internal resources could execute in time `X`. In order to increase the throughput of internal resources, the algorithm calculates the jobs that will be sent to external resources as a function of the jobs that will be submitted to internal resources (lines 28-20). Finally, the algorithm sets the new objectives (lines 21-22), and programs a new event for the next objective update (line 23).

Algorithm 1 only considers one internal resource, and one external resource to calculate the performance of the whole Federated Grid. This algorithm can be easily adapted to a more complex grid with same scenario and more than two participants.

The DO algorithm consists in a few code lines that calculates one linear equation per grid resource to determine the performance of the whole Federated Grid. Thus, it does not need to deploy agents or specialized sensors across the different infrastructures, which is the case for the Network Weather Service [49] or the RPS toolkit [11].

When scheduling, DO behaves like the SO version, that is, it maps no more than `DISPATCH_CHUNK` jobs to resources every `SCHEDULING_INTERVAL` seconds if still there are unscheduled jobs. But in this case, the schedule has to meet the dynamic objective previously calculated.

5.4 Static Objective and Advance Scheduling: SO-AS

The SO-AS strategy is the combination of the SO algorithm plus the AS mechanism. The main difference with the previous schedulers is that AS does not wait for free nodes, instead it queues jobs in advance in the target resources. This is necessary in Federated Grids to avoid latencies and increase performance.

Algorithm 2: Scheduler()

```
1  if (unscheduledJobs.size() == 0) then
2      sendInternalEvent(SCHEDULING_INTERVAL, SCHEDULE_NOW);
3  internalJobs = numInternalJobs(DISPATCH_CHUNK, unscheduledJobs);
4  toInternal = (internalJobs*internalObject)/unscheduledJobs.size();
5  toExternal = internalJobs - toInternal;
6  while ((scheduledJobs < DISPATCH_CHUNK) && (availableInternal || availableExternal)) do
7      j = (Job)it.next();
8      if (j.isInternalJob()) then
9          if (availableInternal && (scheduledToInternal < toInternal)) then
10             if ((availableInternal = scheduleToInternalResource(j)) == true) then
11                 scheduledToInternal++;
12                 scheduledJobs++;
13                 remove();
14                 continue;
15         if (availableExternal && (scheduledToExternal < toExternal)) then
16             if ((availableExternal = scheduleToExternalResource(j)) == true) then
17                 scheduledToExternal++;
18                 scheduledJobs++;
19                 remove();
20     else
21         scheduleToInternalResource(j);
22         scheduledJobs++;
23         remove();
24 dispatch();
25 sendInternalEvent(SCHEDULING_INTERVAL, SCHEDULE_NOW);
```

The details of the scheduler are shown in Algorithm 2. In this pseudo code, the AS algorithm maps no more than `DISPATCH_CHUNK` jobs to resources every `SCHEDULING_INTERVAL` seconds if there are unscheduled jobs (line 1). First, the algorithm determines how many of the `DISPATCH_CHUNK` jobs to schedule are internal jobs (line 3). As we said before, in a Federated Grid, GridWay can receive jobs from internal and external users. Since AS has a defined objective, it has to determine how many of the `internalJobs` jobs to schedule should be submitted to internal resources and how many to external ones. Line 4, `toInternal` shows that it is proportional to number of jobs that the SO algorithm determines that has to be submitted to internal resources. For example, if the static prediction says that 120 of 300 unscheduled jobs should be submitted to internal resources, only 40% of the `internalJobs` jobs can be scheduled to internal resources. The rest of the `internalJobs` jobs are going to be scheduled to external resources (line 5). Then, while `scheduledJobs` jobs keep less than `DISPATCH_CHUNK`, and there are available internal and external resources (line 6), AS maps the next job (line 7). If the job to schedule is in-

ternal (line 8), there are available internal resources, and the internal schedule objective has not been met (line 9), then the job is scheduled to an internal resource (line 10). Basically, `scheduleToInternalResource` queues jobs to a limit of `Number of nodes()*MAX_RUNNING_RESOURCE_ FACTOR`. Otherwise, if there are no available internal resources or the internal schedule objective has been met, AS evaluates line 15. Thus, if there are available external resources and external schedule objective has not been met, the job is scheduled to an external resource. `scheduleToExternalResource` (line 16) behaves like `scheduleToInternalResource`. Otherwise, if the job was not internal (line 20), and in order to avoid the situations on which a participant of the Federated Grid can receive from another one a job previously submitted, GridWay only schedules external jobs to internal resources (line 21). Finally, all the jobs that are being scheduled are dispatched (line 24), and AS programs a new event for the next schedule (line 25).

5.5 Dynamic Objective and Advance Scheduling: DO-AS

The DO-AS strategy incorporates the previously presented DO mechanism to dynamically calculate a new objective by using the r_∞, and $r_{1/2}$ parameters, and the AS algorithm. One again, (see SO-AS case) the main difference with the previous schedulers is that AS does not wait for a free node, instead it queues jobs in advance in the target resources.

6 Design and Implementation

A real infrastructure has previously been set up [48,47] and allows resources from the EGEE infrastructure to be accessed through a GridGateWay. However, the deployment of the proposed algorithms on a real environment will require involvement of a large number of active users and resources, which is very hard to coordinate and build, and would prevent repeatability of results. Simulation appears to be the easiest way to analyze the different proposed mapping strategies. Based on the simulation results, we can later encourage or discourage the deployment on a real production environment. Simulation of a Federated Grid was performed using the GridSim toolkit [21].

6.1 GridWaySim Entities

GridWaySim is the term for our simulation of a Federated Grid. Explanations for the different participating entities of GridWaySim follow:

- **GridWaySim**: this entity represents the complete simulation, and is responsible for the creation of the main simulated entities: GridWay meta-schedulers, Users, Testbeds, and Workloads.
- **GridWay**: the *GridWay* entity represents a generic GridWay meta-sche duler implementing the corresponding algorithm.
- **Testbed**: a *Testbed* represents a generic set of grid resources.
- **User**: the *User* models a user that submits experiments to a GridWay meta-scheduler. We use this entity to represent internal as well as external users. The functionality of each user includes the submission of experiments to the correspondent meta-scheduler, and waiting for it completion.
- **Experiment**: an *Experiment* is a collection of jobs. We use this entity to recover important information about the experiment (such as the start and end times), and all of its jobs.
- **Job**: the *Job* entity represents a generic job submitted to the grid. This entity provides specific information about each job: start time, end time, and CPU time among others. We can represent jobs of different computation times, and with different input and output file sizes.
- **Workload**: the Workload entity submits jobs by reading resource traces from a file. Thus, our jobs are competing with the jobs submitted by the Workload entity [32].

7 Experiments & Results

We have implemented five versions of GridWaySim that only differ in the mapping strategy. These versions are called *Normal*, *SO*, *DO*, *SO-AS*, and *DO-AS*. As a result, we isolate the mapping policy as the only factor that can cause throughput and makespan variations between these five GridWaySim versions. In other words, DO-AS as well as all the previous versions rely on the same configuration, with the same number of users that submit at the same time the same experiment with the same number of jobs (each with the same length, and input and output files size) to the same broker. The number of brokers and resources also remains unchanged.

7.1 Test Scenario

As depicted in Figure 5 we have used the same scenario than in our previous work [31], [30]. There are two grid resources in this test scenario: the DSA (Distributed System Architecture) and the LCG (LHC Computing Grid). The DSA testbed represents the resources of the Distributed System Architecture research group at the Universidad Complutense de Madrid. In the same way, the LCG testbed represents the Large Hadron Collider (LHC) Computing

Grid. From the point of view of a DSA internal user, the DSA GridWay is her broker, the DSA resources are internal resources, and the LCG resources are external resources. In the same way, all the jobs received by the LCG GridWay through the Globus WS-GRAM interface are from external users. This scenario represents a common situation on which a small organization uses the resources of a partner grid infrastructure, but at the same time taking advantage of its own resources.

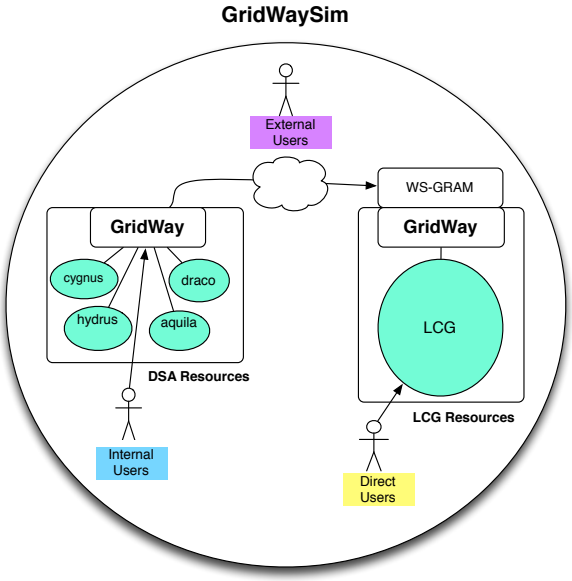

Fig. 5. Test scenario.

Tables 1, and 2 summarize the number of computing elements, aka PEs (Processing Elements), and MIPS (Millions Instructions Per Second) of each machine in the DSA and LCG infrastructures.

Table 1
Characteristics of the machines in the DSA research testbed.

Machine	PEs	MIPS/PE
hydrus	4	9787
aquila	5	9787
orion	1	9787
cygnus	2	6536
draco	1	6536

Table 2
Characteristics of the machines in the LCG research testbed.

Machine	PEs	MIPS/PE
machine0	800	9787
machine1	640	6536
machine2	560	4902

7.2 Simulation Entities and Parameters

When the simulation starts, GridWaySim creates 3 Users, 2 GridWay meta-schedulers, 1 DSATestbed, 1 LCGTestbed, and 1 Workload. Each entity is an independent thread attending petitions in their body() method. The exact configuration of the five versions of GridWaySim follows:

- **GridWay**: since we need to interconnect the DSA and LCG grids to form a federation, we have to instantiate 2 GridWay meta-schedulers, 1 for each grid infrastructure.
- **Testbed**: the resources of the DSA research group are represented by the *DSATestbed* entity, and the LCG ones by the *LCGTestbed* entity. Each follows the configurations depicted in Tables 1 and 2, respectively.
- **Experiment**: every Experiment is a collection of 550 equal Jobs representing a typical sample of medium-sized grid experiment.
- **Job**: the main parameters of each Job are the length or size (in Millions of Instructions, MI) of the Job to be executed, the input files (in bytes), and the output files (also in bytes) to be submitted to the corresponding resource. All Jobs have the same values for the three parameters: the size is 6,000,000 MI, the input file size is 1,000,000 bytes, and the output file size is 2,000,000 bytes.
- **User**: when a User is created, we have to indicate a submit time for her Experiment. Each user only submits one Experiment 48 hours after the previous one. The first User submits her Experiment at 12:00 of the second day of the simulation. Thus, each User submits her experiment to the DSA GridWay at 12:00 of the corresponding day of simulation.
- **Workload**: the Workload entity submits 188,041 jobs to the LCGTestbed at the time specified in the trace file. Workload is the reason why the LCG grid resources might not be available at certain times. The file follows a standard workload format [50]. As a trace file, we used the LCG Grid Log that contains 11 days of real activity from multiple nodes that make up the LCG (Large Hadron Collider Computing Grid [29]). We now enumerate some details about this testbed:
  - **Number of jobs submitted**: 188,041. The log specifies the submit time and the run time of each job.

✻ **Start time**: Sun Nov 20 00:00:05 GMT 2005.
✻ **End time**: Mon Dec 05 10:30:24 GMT 2005.
✻ **Maximum number of machines**: 170.
✻ **Maximum number of computing elements**: 24,515.

Although the number of PEs of the real LCG testbed is 24,515, we decided to reduce the number of PEs in our simulated LCG testbed to those in Table 2, in order to force saturation of LCG resources.

We now provide detailed information about DO and DO-AS mapping strategies parameters:

- *Default Objective*: at the start of the simulation there is not enough information of finished jobs for the DO algorithm to calculate a new objective so we set a default objective as follows. Half of the resources should be submitted to internal resources, and the other half to external resources.
- `OBJECTIVE_INTERVAL`: the DO algorithm is called every 30 seconds.
- `MAX_RUNNING_RESOURCE_FACTOR`: the value of this factor is 3, so if a resource has 5 PEs then AS can queue a maximum of 15 jobs in that resource.
- `SCHEDULING_INTERVAL`: the AS algorithm is called every 30 seconds.
- `DISPATCH_CHUNK`: the AS algorithm schedules 15 jobs each time.

7.3 Results

Tables 3 and 4 summarize the results of the five GridWaySim versions when mapping 550 jobs. We can explain these results based on the level of saturation of the LCG resource, or what is the same, knowing the number of available free PEs. Thus, each User submits her Experiment in a specific instant of the simulation that corresponds with a different LCG saturation level. So, when User-0 submits her Experiment, there is an ideal scenario on which the LCG infrastructure always has free PEs: the *low saturation scenario*. The *medium saturation scenario* is the one suffered by User-1, in this case the LCG resource has fewer free PEs. Finally, User-2 coexists within a *high saturation scenario* of LCG in which there are limited PEs.

In Table 3 we can see the number of jobs that each GridWaySim version executes on both grid infrastructures forming the Federated Grid. In fact, this table shows the objective calculated by SO, DO, SO-AS, and DO-AS. The objectives calculated by the four versions clearly are not very similar to one another. This occurs because the SO and SO-AS versions calculate the objective off-line, all jobs have been finished, thus it has a whole view of the performance of the resources. However, DO, as well as DO-AS calculate a new objective every `OBJECTIVE_INTERVAL` seconds, thus they only have a partial

view of the same performance. Finally, DO differs from DO-AS because of the influence of AS: while AS submits in advance as much jobs as possible at every `SCHEDULING_INTERVAL` seconds, DO only schedules if there are free PEs.

The Normal and DO algorithms behave as expected: as the saturation of LCG increases, more jobs are executed in DSA. SO and SO-AS do not follow this behavior, since as we mentioned before, SO and SO-AS produce poor objectives in high saturation situations because of the long periods on which we cannot execute a single job in LCG. Again, DO-AS behaves in a different way, and calculates very similar objectives for the three levels of saturation. It can be appreciated that only a few more jobs are executed on internal resources for middle, and high saturation situations.

If we combine the results of Table 3 with the completion time achieved by the different GridWaySim versions of Table 4 we obtain a global view of the five algorithms. Since the Normal, SO, and DO algorithms only submit jobs when the resources have free PEs, clearly the results correspond with the different saturation scenarios: as the saturation of LCG increases, more jobs are executed on internal resources, and more time is needed for the Experiment completion. In contrast, since the DO-AS algorithm queues jobs in advance instead of waiting for PEs availability, its performance is more uniform: the number of jobs submitted to each infrastructure and the makespan are almost the same for the three Users. In general, User-0 compared with Users 1 and 2 obtains the best results, but this is obvious since the waiting time on LCG queues is practically null. In addition, for all the Users the DO-AS algorithm

Table 3
Number of jobs executed in each infrastructure for the different GridWaySim versions.

	Normal		SO/SO-AS		DO		DO-AS	
	DSA	LCG	DSA	LCG	DSA	LCG	DSA	LCG
User-0	26	524	102	448	31	519	67	483
User-1	113	437	145	405	124	426	70	480
User-2	427	123	124	426	416	134	70	480

Table 4
Completion time achieved for the different GridWaySim versions.

	Normal	SO	DO	SO-AS	DO-AS
User-0	1:37:36	1:35:39	1:36:31	1:47:27	1:30:21
User-1	2:06:40	2:19:31	2:05:23	2:18:03	1:37:55
User-2	6:18:41	13:27:13	6:12:49	2:05:41	1:38:41

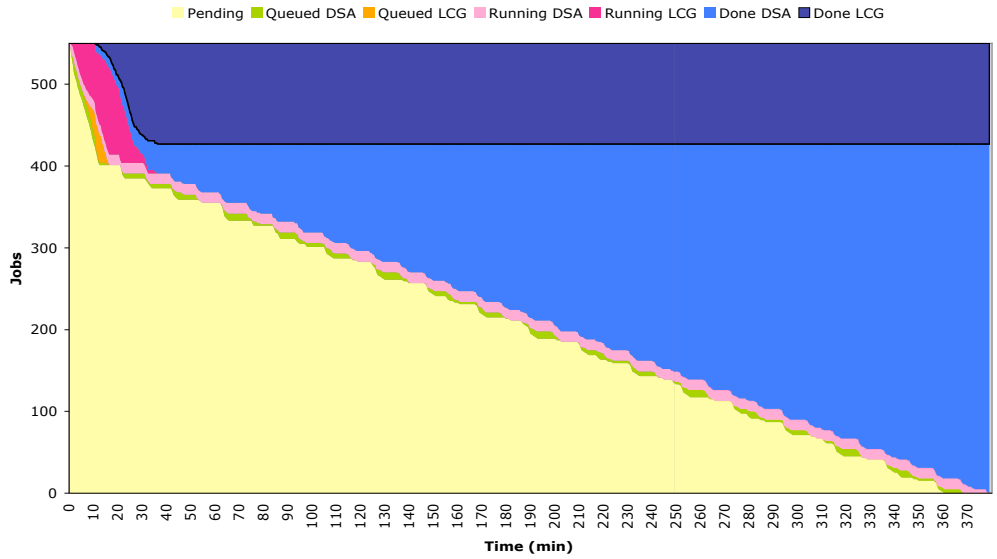

Fig. 6. Jobs state over time by using the *Normal* algorithm on DSA, LCG, and Federated Grid infrastructures for User-2 (LCG highest saturation situation).

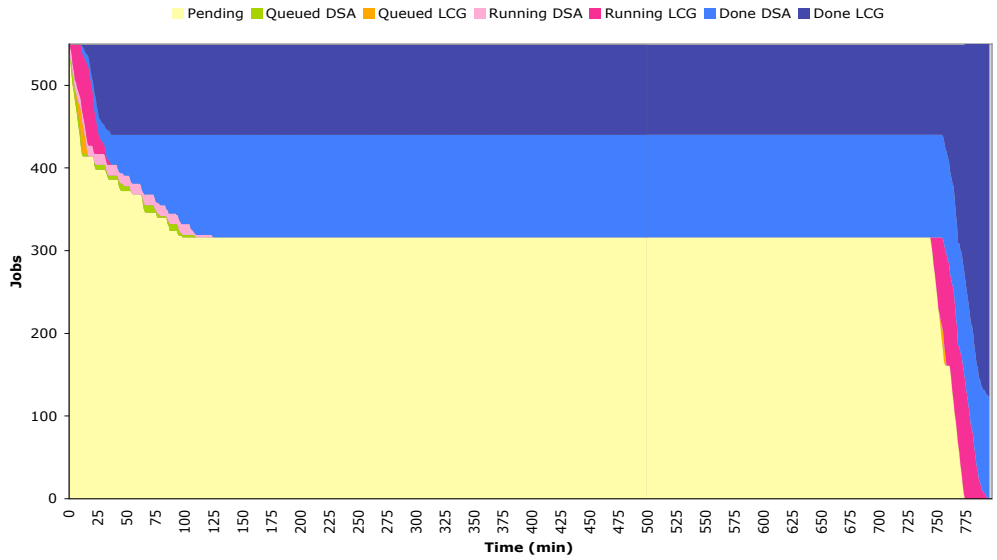

Fig. 7. Jobs state over time by using the *SO* algorithm on DSA, LCG, and Federated Grid infrastructures for User-2 (LCG highest saturation situation).

always reduces the makespan obtained by the previous algorithms. The most remarkable reduction obtained by DO-AS is for User-2. This reduction is due to the queuing of jobs on resources, instead of waiting for free PEs as the rest of the algorithms do.

Finally, the successive versions, except SO, and SO-AS for high saturation situations, improve the results obtained by the previous ones in terms of through-

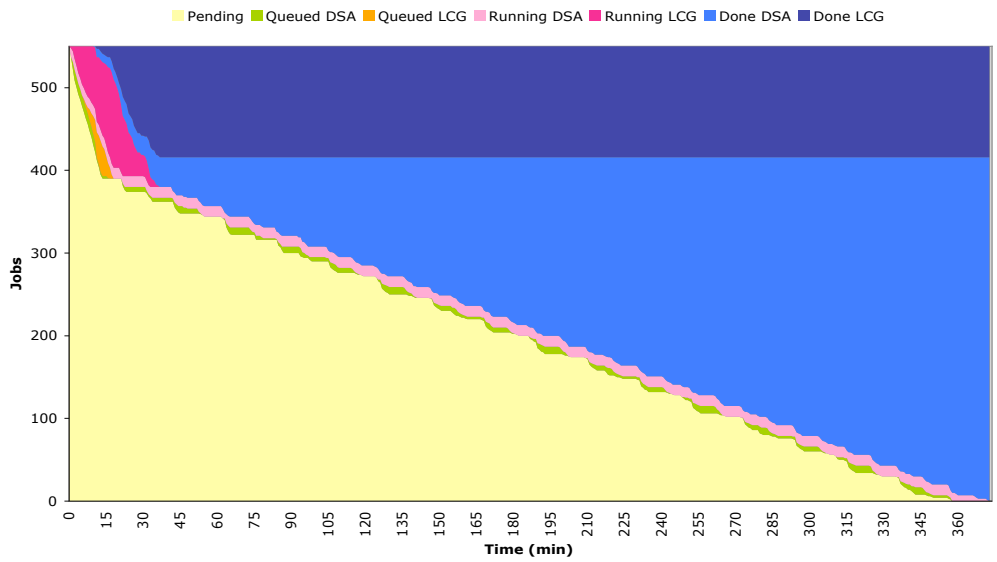

Fig. 8. Jobs state over time by using the *DO* algorithm on DSA, LCG, and Federated Grid infrastructures for User-2 (LCG highest saturation situation).

put, and completion time. Thus, the combination of the performance model with the scheduling in advance has reveal as the best approach for mapping jobs in Federated Grids.

In Figures 6, 7, 8, 9, and 10 we can graphically see, through the state of the jobs over the time, the difference in the behavior between the five algorithms for User-2, which represents LCG highest saturation situation. We can divide the graphics in two groups. In one group we place the graphics of the Normal, SO, and DO algorithms that present a similar behavior, mostly in the queue regions that are very small.

In the second group we include the SO-AS and DO-AS strategies since their queued regions are similar. A closer analysis demonstrates that with the SO-AS strategy jobs are pending during more time, and more jobs are queued in internal resources while with the DO-AS algorithm more jobs are queued in external resources, and jobs remain in a pending state for less time.

7.4 Comparison

Summing up, SO compared with the Normal version increases the throughput of internal resources, except for User-2: SO increases the number of jobs executed per time unit. However, SO does not reduce the makespan achieved by the Normal version. We have observed that the performance model does not fit well in saturation situations. In these cases, there are long periods on

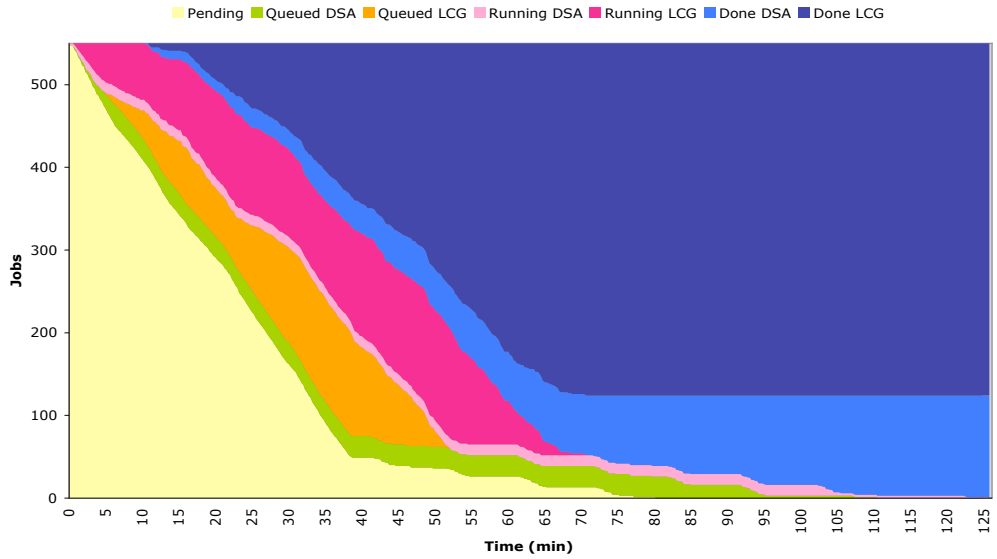

Fig. 9. Jobs state over time by using the *SO-AS* algorithm on DSA, LCG, and Federated Grid infrastructures for User-2 (LCG highest saturation situation).

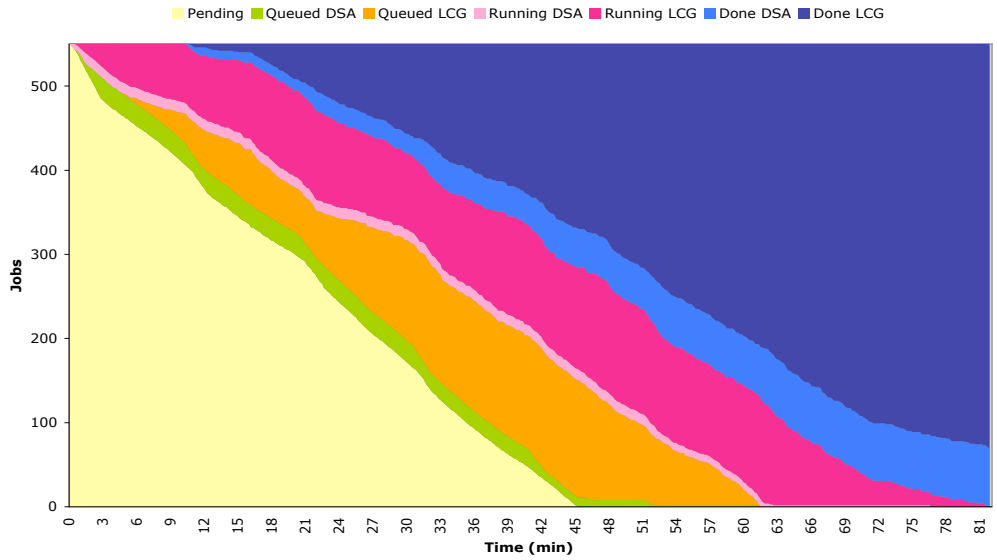

Fig. 10. Jobs state over time by using the *DO-AS* algorithm on DSA, LCG, and Federated Grid infrastructures for User-2 (LCG highest saturation situation).

which there are no available nodes for the execution of jobs. A linear equation cannot properly adapt to the graphics produced by this type of behavior. As a consequence, the calculated objective is not correct, and produces poor results.

When DO is compared with the Normal version, an increase in the throughput of internal resources occurs, and spends less time to complete all the jobs.

Compared with the SO version, DO completes all jobs in less time, and solves the problem detected in saturation situations. The overall reduction in the makespan is not really significant.

SO-AS, compared with the static and dynamic versions only reduces the makespan in the case of high saturation in the external resources.

Finally, DO-AS, compared with all the previous versions, reduces the makespan in all cases: while DO-AS schedules based on the *performance* of the different infrastructures forming the Federated Grid, the previous algorithms schedule only considering the *states* of the resources. Thus, the combination of the calculation of a dynamic objective plus the advance scheduling provides the best results for all algorithms.

8 Conclusions and Future Work

In this paper we have presented a decentralized model to solve the problem of scheduling in Federated Grids. Our approach provides an alternative to centralized, application-centric or ad-hoc solutions. We have proposed the deployment of GridWay as the meta-scheduler on each grid infrastructure of the Federated Grid. Four possible algorithms have also been presented that could be executed in the GridWay meta-scheduler. Experimental results and graphics are provided to compare the behavior of the different policies. Our analysis reveals that DO-AS is the best strategy. The DO-AS mechanism for mapping jobs on the Federated Grid is completely transparent for users and applications. Each DO-AS at each Grid infrastructure of the Federated Grid is aware of reduction of the makespan of its own applications and increasing the performance of its own Grid infrastructure. DO-AS does not require information on processor speeds nor task lengths. The algorithm only needs information about the past performance of the resources to predict a new objective. Also, DO-AS does not need to deploy agents or specialized sensors across the different infrastructures, which is the case for the Network Weather Service or the RPS toolkit. In addition, the calculation of a new objective is an extremely light process. Thus, DO-AS is fast enough to be used in realistic scheduling scenarios. Finally, DO-AS is a decoupled solution on which schedulers at different levels are independent: local schedulers, as well as workload managers do not need to modify their policies. This additional flexibility allows us to deploy our algorithm in any Grid.

We are currently testing our algorithms in a simulated testbed formed by several grid infrastructures. Improvements in the algorithms are also under study, specifically by adapting the values of important parameters, such as the suspension time out, at runtime. This parameter determines the maxi-

mum suspension time (in seconds) in the local job management system. When this value is exceeded the job is migrated to another host. The value of this parameter depends on the applications length and resource saturation. In addition, we want to investigate the effects of sharing policies in the proposed mapping policies. Finally, we will implement the best algorithm in GridWay for its deployment in a real infrastructure.

References

[1] A. Agarwal, M. Ahmed, A. Berman, B.L. Caron, A. Charbonneau, D. Deatrich, R. Desmarais, A. Dimopoulos, I. Gable, L.S. Groer, R. Haria, R. Impey, L. Klektau, C. Lindsay, G. Mateescu, Q. Matthews, A. Norton, W. Podaima, D. Quesnel, R. Simmonds, R.J. Sobie, B. St. Arnaud, C. Usher, D.C. Vanderster, M. Vetterli, R. Walker, and M. Yuen. "GridX1: A Canadian computational grid". *Future Generation Computer Systems*, 23(5):680–687, June 2007.

[2] Alain Andrieux, Dave Berry, Jon Garibaldi, Stephen Jarvis, Jon MacLaren, Djamila Ouelhadj, and Dave Snelling. "Open Issues in Grid Scheduling". Technical Report ISSN 1751-5971, UK e-Science Institute, October 2003.

[3] Francine Berman. *"The Grid: Blueprint for a Future Computing Infrastructure"*, chapter "High-Performance Schedulers". Morgan Kaufmann, 1998.

[4] Francine Berman, Richard Wolski, Henri Casanova, Walfredo Cirne, Holly Dail, Marcio Faerman, Silvia Figueira, Jim Hayes, Graziano Obertelli, Jennifer Schopf, Gary Shao, Shava Smallen, Neil Spring, Alan Su, and Dmitrii Zagorodnov. "Adaptive Computing on the Grid Using AppLeS". *IEEE Transactions on Parallel and Distributed Systems*, 14:369–382, 2003.

[5] Richard Blake, Peter V. Coveney, Peter Clarke, and S. M. Pickles. "The Teragyroid Experiment - Supercomputing 2003". *Scientific Programming*, 13(1):1–17, 2005.

[6] Bruce Boghosian, Peter Coveney, Suchuan Dong, Lucas Finn, Shantenu Jha, George Karniadakis, and Nicholas Karonis. "NEKTAR, SPICE and Vortonics: using federated grids for large scale scientific applications". In *Proceedings of the 2006 IEEE Challenges of Large Applications in Distributed Environments*, pages 34–42. IEEE Computer Society Press, 2006.

[7] Tracy D. Braun, Howard Jay Siegel, Noah Beck, Ladislau Blni, Muthucumaru Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, and Bin Yao. "A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems". *The 17th IEEE Symposium on Reliable Distributed Systems*, pages 330–335, 1998.

[8] Thomas L. Casavant and Jon G. Kuhl. "A taxonomy of scheduling in general-purpose distributed computing systems". *IEEE Transactions on Software Engineering*, 14(2):141–154, 1988.

[9] The Distributed ASCI Supercomputer 2 (DAS-2). http://www.cs.vu.nl/das2/, 2008.

[10] Marcos Dias de Assuncao, Rajkumar Buyya, and Srikumar Venugopal. "InterGrid: A Case for Internetworking Islands of Grids". *Concurrency and Computation: Practice and Experience (CCPE)*, 20:997–1024, July 2007.

[11] Peter A. Dinda. "Design, implementation, and performance of an extensible toolkit for resource prediction in distributed systems". *Parallel and Distributed Systems, IEEE Transactions on*, 17(2):160–173, 2006.

[12] Fangpeng Dong and Selim G. Akl. "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems". Technical Report 2006-504, Ontario Queens University, January 2006.

[13] Enabling Grids for E-sciencE (EGEE). http://www.eu-egee.org, 2008.

[14] Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. "Benefits of Global Grid Computing for Job Scheduling". In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, pages 374–379, 2004.

[15] The UK e-Science Programme. http://www.rcuk.ac.uk/escience, 2008.

[16] I. Foster, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer. "OGSA Basic Execution Service". Technical Report GFD-R.108, Open Grid Forum, November 2008.

[17] Ian Foster. "What is the Grid? A Three Point Checklist". *GRIDtoday*, 1(6), 2002.

[18] Philip W. Fowler, Shantenu Jha, and Peter V. Coveney. "Grid-based steered thermodynamic integration accelerates the calculation of binding free energies ". *Philosophical Transactions of The Royal Society*, 363(1833):1999–2015, August 2005.

[19] Richard F. Freund, Michael Gherrity, Stephen Ambrosius, Mark Campbelly, Mike Halderman, Debra Hensgenz, Elaine Keithy, Taylor Kiddz, Matt Kussowy, John D. Limay, Francesca Mirabilea, Lantz Moorex, Brad Rusty, and H. J. Siegel. "Scheduling Resources in Multi-User, Heterogeneous, Computing Environments with SmartNet ". In *Proceedings of the 7th International IEEE Heterogeneous Computing Workshop (HCW'98)*, pages 3–19, 1998.

[20] Noriyuki Fujimoto and Kenichi Hagihara. "A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks". In *Proceedings of the 2004 International Symposium and the Internet Workshops (SAINTW'04)*, pages 674–680, 2004.

[21] GridSim Toolkit. http://www.gridbus.org/gridsim/, 2008.

[22] The Globus Project. http://www.globus.org, 2008.

[23] Francesc Guim and Julita Corbalan. "A Job Self-Scheduling Policy for HPC Infrastructures". In *Proceedings of the 13th Workshop on Job Scheduling Strategies for Parallel Processing*, volume 4942/2008, pages 51–75, 2007.

[24] R.W. Hockney and C.R. Jesshope. *"Parallel Computers 2: Architecture, Programming, and Algorithms"*. Adam Hilger Ltd, 1988.

[25] Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. "A Framework for Adaptive Execution on Grids". *Software – Practice and Experience*, 34(7):631–651, 2004.

[26] Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. "A recursive architecture for hierarchical grid resource management". *Future Generation Computer Systems*, 25(4):401–405, April 2009.

[27] Peter Kacsuk, Tamas Kiss, and Gergely Sipos. "Solving the grid interoperability problem by P-GRADE portal at workflow level". *Future Generation Computer Systems*, 24(7):744–751, July 2008.

[28] Attila Kertesz, Ivan Rodero, and Francesc Guim. "Meta-Brokering requirements and research directions in state-of-the-art Grid Resource Management". In *Proceedings of the CoreGRID Integration Workshop 2008*, pages 371–382, April 2008.

[29] Worldwide LHC Computing Grid. http://lcg.web.cern.ch/LCG/, 2008.

[30] Katia Leal, Eduardo Huedo, and Ignacio M. Llorente. "Dynamic Objective and Advance Scheduling in Federated Grids". In *Proceedings of the On the Move to Meaningful Internet Systems: OTM 2008*, volume 5331/2008, pages 711–725. Springer Berlin / Heidelberg, November 2008.

[31] Katia Leal, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. "Scheduling Strategies in Federated Grids". In *Proceedings of the 2008 High Performance Computing & Simulation Conference (HPCS 2008)*, pages 117–123. ECMS, 2008.

[32] Hui Li and Rajkumar Buyya. "Model-based simulation and performance evaluation of grid scheduling strategies". *Future Generation Computer Systems*, 25(4):460–465, April 2009.

[33] Ignacio M. Llorente, Rubén S. Montero, Eduardo Huedo, and Katia Leal. "A Grid Infrastructure for Utility Computing". In *Proceedings of the Third International Workshop on Emerging Technologies for Next-generation GRID (ETNGRID 2006)*, pages 163–168. IEEE Computer Society Press, 2006.

[34] Load Sharing Facility. http://www.platform.com/Products/platform-lsf, 2008.

[35] Rubén S. Montero, Eduardo Huedo, and Ignacio. M. Llorente. "Benchmarking of High Throughput Computing Applications on Grids". *Parallel Computing*, 32(4):267–269, 2006.

[36] Portable Barch System. http://www.pbsgridworks.com/, 2008.

[37] Platform Computing, The Evolution Of Grid: The Three Stages of Grid Computing. http://www.platform.com/grid/evolution.asp, 2007.

[38] Christopher Pinchak, Paul Lu, and Mark Goldenberg. *"Job Scheduling Strategies for Parallel Processing"*, chapter " Practical Heterogeneous Placeholder Scheduling in Overlay Metacomputers: Early Experiences ", pages 205–228. Springer Berlin, 2002.

[39] The German Grid Initiative (D-Grid). http://www.d-grid.de, 2008.

[40] Distributed Resource Management Application API. https://forge.gridforum.org/projects/drmaa-wg, 2008.

[41] Globus Toolkit 4.0 WS_GRAM. http://www.globus.org/toolkit/docs/4.0/execution/wsgram/, 2008.

[42] Gerald Sabin, Rajkumar Kettimuthu, Arun Rajan, and Ponnuswamy Sadayappan. "Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment". In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2862, pages 87–104, 2003.

[43] Bianca Schroeder and Mor Harchol-Balter. "Evaluation of Task Assignment Policies for Supercomputing Servers: The Case for Load Unbalancing and Fairness". *Cluster Computing*, 2004.

[44] Sun Grid Engine. http://www.sun.com/software/gridware/, 2008.

[45] The TeraGrid Project. http://www.teragrid.org/, 2008.

[46] Jeffrey D. Ullman. "NP-Complete Scheduling Problems". *Journal of Computer and System Sciences*, 10(3):384–393, 1975.

[47] Constantino Vázquez, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. "Evaluation of A Utility Computing Model based on Federation of Grid Infrastructures". In *13th International Euro-Par Conference (Euro-Par 2007)*, volume 4641/2007, pages 372–381. Lecture Notes in Computer Science (LNCS), august 2007.

[48] Constantino Vázquez, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. "A Performance Model for Federated Grid Infrastructures". In *Proceedings of the 16th Euromicro International Conference on Parallel, Distributed and network-based Processing (PDP 2008)*, pages 188–192, 2008.

[49] Rich Wolski, Neil T. Spring, and Jim Hayes. "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing". *Future Generation Computer Systems*, 15(5–6):757–768, 1999.

[50] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, 2008.

[51] Hao Yin, Huilin Wu, and Jiliu Zhou. "An Improved Genetic Algorithm with Limited Iteration for Grid Scheduling". In *Proceedings of the Sixth International Conference on Grid and Cooperative Computing (GCC 2007)*, pages 221–227, 2007.

[52] Jianhui Yue. "Global Backfilling Scheduling in Multiclusters". In *Proceedings of the Second Asian Applied Computing Conference (AACC)*, volume 3285/2004, pages 232–239, 2004.