A Simulation-based Scheduling Strategy for Scientific Workflows

Sergio Hernández, Javier Fabra, Pedro Álvarez and Joaquín Ezpeleta
Aragón Institute of Engineering Research (I3A), Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza, Spain

Keywords: Scientific Workflows, Grid Modelling and Simulation, Workloads, Performance Analysis.

Abstract: Grid computing infrastructures have recently come up as computing environments able to manage heterogeneous and geographically distributed resources, being very suitable for the deployment and execution of scientific workflows. An emerging topic in this discipline is the improvement of the scheduling process and the overall execution requirements by means of simulation environments. In this work, a simulation component based on realistic workload usage is presented and integrated into a framework for the flexible deployment of scientific workflows in Grid environments. This framework allows researchers to simultaneously work with different and heterogeneous Grid middlewares in a transparent way and also provides a high level of abstraction when developing their workflows. The approach presented here allows to model and simulate different computing infrastructures, helping in the scheduling process and improving the deployment and execution requirements in terms of performance, resource usage, cost, etc. As a use case, the Inspiral analysis workflow is executed on two different computing infrastructures, reducing the overall execution cost.

1 INTRODUCTION

Grid computing emerged as a paradigm for the development of computing infrastructures able to share heterogeneous and geographically distributed resources (Foster and Kesselman, 2003). Due to their computational and networking capabilities, this type of infrastructure has turned into execution environments suitable for scientific workflows, which require intensive computations as well as complex data management. Nevertheless, the comparison of existing Grid workflow systems has shown relevant differences in the building and execution of workflows that causes experiments programmed by scientists and engineers to be strongly coupled to the underlying system responsible for their execution (Rahman et al., 2011; Yu and Buyya, 2005). Therefore, two of the most interesting open challenges in the field of scientific computing are the ability to program scientific workflows independently of the execution environment and the flexible integration of heterogeneous execution environments to create more powerful computing infrastructures for their execution.

This new generation of computing infrastructures requires new strategies of resource brokering and scheduling to facilitate the utilization of multiple-domain resources and the allocation and binding of workflow activities to them. An emerging topic in this discipline is the use of simulation environments to help in the scheduling process, improving the overall execution requirements in terms of resource usage, time and costs. Some approaches such as GMBS (Kertész and Kacsuk, 2010) or SCI-BUS [1], for instance, propose the use of simulation tools to evaluate the best meta-scheduling strategy. Different scheduling policies can be evaluated to decide the most suitable allocation of workflow activities to resources. On the other hand, another research focus on the development of a novel scheduling algorithm and its execution over a simulated environment. The results are then compared with other similar algorithms in order to classify the algorithm with respect to some predefined criteria. Strategies are normally compared in terms of makespan (Hamscher et al., 2000; Abraham et al., 2006; Yu and Shi, 2007), simulation times (Ludwig and Moallem, 2011) or queue times (Yu and Shi, 2007; Ludwig and Moallem, 2011).

Regardless of the problem to be solved, simulation environments may consider execution environment models and workloads with the purpose of improving scheduling decisions. The first provide a complete specification of architectures and configurations of the execution environment. Flexible mechanisms for the specification of these models should be

[1] http://www.sci-bus.eu/

Hernández S., Fabra J., Álvarez P. and Ezpeleta J..
A Simulation-based Scheduling Strategy for Scientific Workflows.
DOI: 10.5220/0004061200610070
In *Proceedings of the 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2012), pages 61-70
ISBN: 978-989-8565-20-4

provided, specially to model evolving and heterogeneous computing infrastructures. Meanwhile, workloads are logs of job sets based on historical data or statistical models representing jobs to be executed in the environment. The relation between workloads and scheduling policies turns around the necessity of using a workload fitting the characteristics of jobs executed in the infrastructure in order to evaluate the suitability of a concrete scheduling algorithm in real terms. In (Feitelson, 2002), the benefits of using workloads as well as how to use them to evaluate a system are discussed. However, their use is still rather limited, due mainly to the complexity of its creation, being the process automation a difficult task. Therefore, workloads are mainly used just for the analysis of Grid systems (Iosup and Epema, 2011; Li et al., 2004). Understanding these real workloads is a must for the tuning of existing Grids and also for the design of future Grids and Cloud infrastructures.

In (Fabra et al., 2012), a framework for the deployment and execution of scientific workflows whose main features are described in Section 2 was presented. This framework facilitates the flexible integration of heterogeneous Grid computing environments, addressing the challenge of creating more powerful infrastructures. Besides, its architectural design guarantees that workflow programmers do not need to be aware of this heterogeneity. In this paper, we integrate new components into our framework for the simulation of scientific workflows using realistic workloads, allowing the improvement and flexibility of job allocation by means of a meta-scheduler. Unlike other approaches which are focused on assisting the researcher, in our proposal simulation results are internally used to make scheduling decisions transparently to researchers and their workflows. Obviously, the complexity of this simulation-based scheduling is increased by the evolving nature of the underlying computing infrastructure.

The information obtained from the simulator component can also be used by the meta-scheduler in order to carry out some optimization process depending on the parameters to be optimized. For instance, it is possible to provide a better-execution-time algorithm which schedules the execution of jobs on the most suitable computing infrastructure depending on the workload provided at the execution time. It is also possible to easily minimize resource costs while keeping a defined relation between execution time and involved costs, for instance.

The remainder of this paper is organized as follows. The main features of the developed framework in which the presented simulation approach is integrated are described in Section 2. The design and implementation of the simulator is sketched by means of the application to a real cluster which uses Condor in Section 3. The flexibility and reuse capabilities of the component are then depicted in Section 4 by means of the integration of another real Grid managed by gLite. Then, the simulation approach integration is applied to the development of a real case study, the LIGO Inspiral analysis workflow in Section 5. Finally, Section 6 concludes the paper and addresses future research directions.

2 EVOLVING TOWARDS THE ADAPTABLE DEPLOYMENT OF SCIENTIFIC WORKFLOWS

The proposed Grid-based framework for programming and executing scientific workflows is able to tackle some of the open challenges in the field of Grid computing. From the programmer's point of view, workflows can be programmed independently of the execution environment where the related tasks will be executed. Different standard languages, widely accepted by the scientific community (e.g. Taverna), can be used for programming this type of abstract workflows. On the other hand, the proposed framework is open and flexible from the computing resource integration's point of view. First, and in accordance with this feature, it is able to simultaneously work with different Grid middlewares or middlewares implemented using other alternatives (e.g. Web services). And, secondly, heterogeneous execution environments can be added, modified or even removed without previous announcement and in a flexible and transparent way. Therefore, the combination of these features turns our solution into a novel and suitable proposal in the field of scientific workflows (Yu and Buyya, 2005).

Figure 1 shows the high-level architecture of the proposed framework. A more detailed description is outside the scope of this paper. Let us concentrate on the process of executing workflow tasks and the architectural components involved in it.

Once a workflow has been deployed, the *workflow execution environment* is responsible for controlling its execution and submitting tasks to the *resource broker* by means of its interface as they must be executed. Submitted tasks are then stored into the *message repository* as messages that encapsulate the information needed for the execution of a task, including the application to be executed, the references to input and output data, a description of the resources required for its execution (operating system, CPU ar-

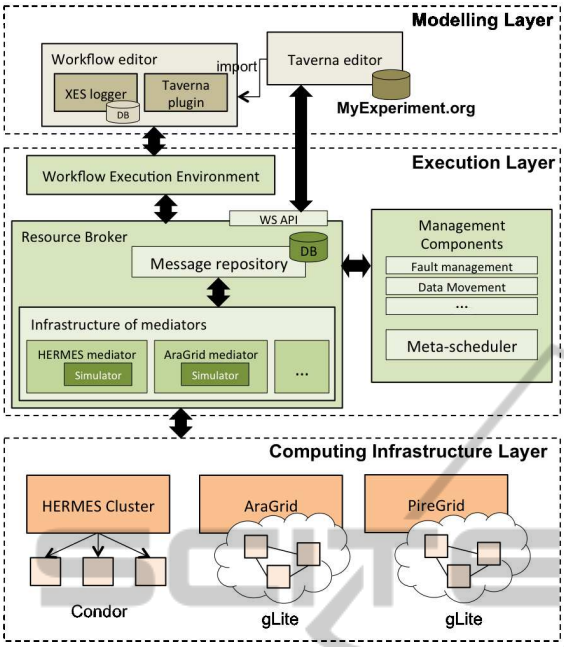

Figure 1: Architecture of the execution environment.

chitecture and features, memory, network bandwidth, etc.) and QoS parameters. These messages are described using the JSDL standard. Optionally, the target computing environment responsible for the task execution can be also included into the message. This type of tasks is called *concrete tasks*. Nevertheless, workflows will be usually programmed independently of the execution environment where their tasks will be executed (*abstract tasks*). This decision tries to take full advantage of the integration capabilities of rid-based framework.

An *infrastructure of mediators* uncouples the resource broker from the specific and technological details about the Grid-based computing environments where tasks will be executed. Each computing environment is represented by a mediator. Internally, a mediator handles a complete information about the Grid infrastructure it represents. Subsequently, this knowledge will be used by the mediator to interact with the message repository and to find at run-time abstract tasks that could be executed by its middleware. Therefore, mediators are responsible for making dispatching decisions related to the execution of tasks. Obviously, in this dispatching model more than one mediator could compete for the execution of a specific task (the criterion would be that their corresponding middlewares were able to execute it). This proposal is an alternative to traditional solutions based on the use of a centralized task scheduler responsible for deciding where tasks will be executed.

Finally, each mediator dispatches its tasks to the middleware managing the infrastructure it represents for their execution and stores the results of the executed tasks into the message repository, as well as the resulting execution log, which can be used for monitoring or analysis purposes. These results will be subsequently recovered by the workflow execution environment for controlling the execution of the deployed workflow.

2.1 Improving the Scheduling Capabilities of the Framework

The dispatching strategy of our proposal presents a set of drawbacks: 1) performance issues related to the execution of tasks are not considered by mediators (therefore, a task could be executed by an inappropriate computing environment degrading the performance of the whole workflow); 2) dispatching decisions are locally adopted by each mediator and, consequently, one of them could monopolize the execution of pending tasks (this could cause unnecessary overloads on its corresponding computing environment); and, finally, 3) the real behaviour of the existing computing environments and the state of their resources is also ignored by the mediators.

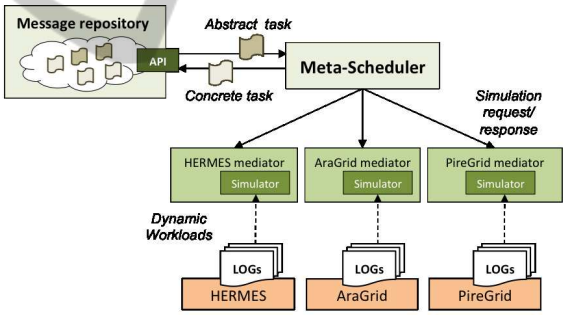

Figure 2: Architectural components for the simulation-based scheduling.

In order to solve the previous drawbacks and also to improve and enhance our infrastructure, a *meta-scheduler* based on simulation techniques will be integrated into the Grid-based framework in this paper. Figure 2 represents the alternative process of executing workflow tasks using a meta-scheduler. Initially, pending (abstract) tasks are stored into the message repository. The meta-scheduler retrieves this type of tasks for determining where they will be finally executed. Scheduling decisions are made by simulating the execution of each task in the existing computing environments and analysing the simulation results. With these results, the task is made concrete and then submitted to the message repository, allowing the task to be executed by the selected mediator.

The interface of mediators has been extended to support this process. Now, each mediator exposes a set of operations able to simulate the execution of a task. Internally, a simulator has been integrated into each mediator for providing the required functionality. More specifically, the simulator is able to: 1) model the corresponding computing environment managed by the mediator (computing resources, memory, network bandwidth, user and scheduling internal policies, etc.); 2) select the most suitable workload for representing the real behaviour of the computing environment and the state of its resources (execution logs are used for creating these workloads); and, finally, 3) simulate the execution of tasks measuring parameters such as the execution time, the data transfer time, the queuing time, the consumed memory, etc.

In the following, the design and implementation of the simulator component is depicted. As it will be shown, this component is flexible enough as to allow an easy adaptation for different computing infrastructures with different scheduling policies.

3 SIMULATING WORKFLOW'S EXECUTION

As stated, the simulator component has been integrated as an internal component in each mediator. Therefore, each computing infrastructure can handle different and customized simulation capabilities. Anyway, simulators are accessed through a well defined API, so adding new simulators to the framework is a guided and easy process. Also, coupling simulation components with mediators allows developers to introduce new computing infrastructures without needing to implement them. Obly the corresponding scheduling policy and the associated simulator must be considered.

The simulation component receives the Grid model and the workload as an input, which are stored as files accessible from the corresponding mediator. Then, after a processing cycle, it generates as a result the execution estimation in terms of time and resource usage with respect to the input provided. The simulator also provides some metrics for analysis purposes such as the average system utilization of each resource, for instance, which can be used to improve the process.

In the following, the design and implementation of the simulation component is sketched by means of the description of two real use cases: the HERMES cluster and the AraGrid multi-cluster Grid.

3.1 Overview of the HERMES Cluster

HERMES is a cluster hosted by the Aragón Institute of Engineering Research (I3A)[2]. In general terms, HERMES consists of 1308 cores and 2.56 TB of RAM. More specifically, it consists of 126 heterogeneous computing nodes, including 52 nodes with two 2.33 GHz 4-core Intel Nehalem CPUs and 24 GB of RAM per node, 48 nodes with two 2.00 GHz 8-core AMD Magny-Cours CPUs and 16 GB of RAM per node, 12 nodes with a 3.00 GHz 4-core Intel Woodcreest quadcore CPUs and 8 GB of RAM per node, 11 nodes with two 2.33 GHz 2-core Intel Woodcreest CPUs and 4 GB of RAM per node, and 4 nodes with two 2.66 GHz 4-core Intel Woodcrest CPUs and 16 GB of RAM per node. The computing nodes in HERMES are connected by Gigabit links, allowing high-speed data transfers.

At the moment of this writing, the cluster is managed by the Condor[3] middleware version 7.6.3.

The cluster is used by a vast variety of researchers, mainly focused on inductive and physical systems, automotive systems, discrete event system analysis and complex semantic workflow analysis. System utilization is usually focused on the use of CPUs rather than memory consumption. Data inputs are usually small sized, although there is a group handling complex experiments with files of more than 20TB. The analysis of relevant workloads shown that the average user is not aware of load peaks or advanced configuration issues, which normally produces that experiments last extremely long, require oversized resources or even are queued for long times. In this scenario, our proposal for a framework which would optimize such situations is extremely useful from both the researcher and also the system usage perspectives.

3.2 Implementation Details of the HERMES Simulator

Alea (Klusáček and Rudová, 2010) has been used to implement the internal simulator in the HERMES mediator component. Alea is an event-based simulator built upon the GridSim toolkit (Sulistio et al., 2008). Alea extends GridSim and provides a central scheduler, extending some functionalities and improving scalability and simulation speed. Alea has been designed to allow an easy incorporation of new scheduling policies and to easily extend its functionalities. Also, Alea provides an experimentation environment easy to configure and use, which helps in the quick

[2] http://i3a.unizar.es

[3] http://research.cs.wisc.edu/condor/

and rapid development of simulators when a new infrastructure is going to be added to the system.

The original implementation of Alea has been extended to allow some Condor features such as user priorities, RAM requirements and preemptions. Figure 3 depicts the structure of the simulator. As shown, it consists of two input files, the *workload* and the *Grid model*, and four main modules, the *Job Loader*, the *Machine Loader*, the *Scheduler* and the *Result Collector*, respectively.

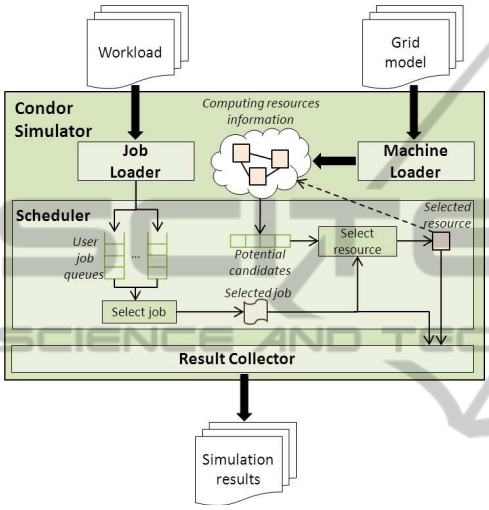

Figure 3: Architecture of the Condor simulator based on Alea.

Multiple workload have been composed using the cluster execution logs from the last year and identifying common situations of resource utilization and job submission. The workload is represented using the Grid Workload Format (GWF format) proposed by the Grid Workload Archive (GWA) (Iosup et al., 2008a). For each job, the job execution time, the number of CPUs required, the memory requirements, the user and group who executes the job and the job dependencies (if exists) are provided. More details on the creation of workloads is provided in subsection 5.1.

The *Grid model* is a text file that contains the information of each computing node. The representation of each node includes a node identifier, the number of machines, the number of CPUs per machine, the total amount of memory per machine, the system architecture, its operating system and the network characteristics. Also, a failure model can be detailed to reflect dynamic changes in the infrastructure during the simulation.

The *Job Loader* component reads the job descriptions and sends them to the scheduler. This module has been extended to allow RAM requirements and user and group details of the submitted jobs.

The *Machine Loader* component is responsible for reading the resource description from a file containing the Grid model. This module has been extended to be able to parse and save the information provided.

The *Scheduler* component is the more complex one. It has been extended with a new scheduling policy considering the schema for user priorities that Condor applies in HERMES. This scheduling policy works as follows: when a job sent by the Job Loader reaches the scheduler, the job is queued in the right user queue. This queue is ordered by the job priority and the job arrival time. When the scheduler requests a new job to be executed, jobs are ordered by their user priority and the job with the highest priority is chosen. Then, the machines with available resources (CPUs and RAM) and also the machines that could have available resources (if some running jobs are evicted) are selected as potential candidates to execute the job. The list of all potential candidates is ordered by multiple criteria (job preferences, machine preferences, etc.) to get the most suitable resource. If there is no resource available to execute the job, this is queued again and the scheduler looks for the next job. Finally, when a job and a resource have been chosen, the job is sent to the resource and its state is updated. In addition, some of the current running jobs are evicted from the selected resource if necessary to execute the new job. These evicted jobs are requeued and will be reexecuted later.

Finally, the Result Collector component is responsible for storing the simulation results and provide them as output. When a job is sent to a resource, evicted or a machine fails, the Result Collector stores this information. When a job ends, the Result Collector stores the job information in an output file. For each job, the arrival time, the time the job has spent queued, the execution time of the resource, the resource where the job was executed and the number of evictions suffered by the job are stored in the file.

3.3 Validation of the HERMES Simulator

The aim of the developed simulator is to be used as a decision tool at meta-scheduling level. In terms of simulation accuracy, its validation is a key issue to verify its feasibility and usefulness for this purpose (Sargent, 2010). Figure 4 shows a comparison of the actual cluster utilization, extracted from the logs, and the simulated utilization, obtained from the simulation of the tasks described in the workload. The comparison is presented as a daily cycle in which the horizontal axis indicates the time (in hours) and the ver-

tical axis shows the CPU utilization rate (in percentage). As it can be observed, the simulation results are very similar to real results. Both plots follow the same trend, being the simulation utilization slightly lower. In terms of the deviation of the simulation results, an average error of 15.09% and a standard deviation of 8.03% is observed.

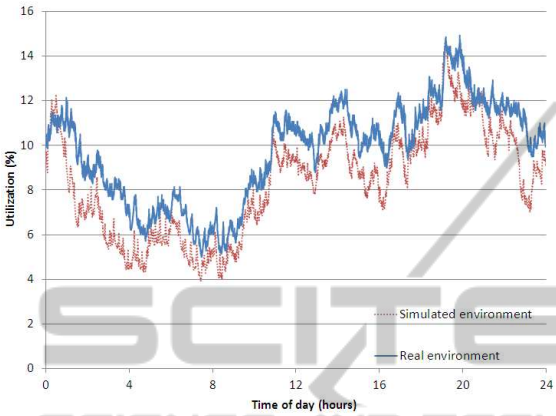

Figure 4: Condor cluster utilization for the real and simulated environment.

In order to validate the job performance indicator, two metrics are provided: the execution time and the queue time. Figure 5 shows the cumulative distribution function for the execution time (Figure 5-a) and the queue time (Figure 5-b). For the sake of clarity, the horizontal axis is shown on a log scale. Figure 5-a illustrates that job execution time is almost the same in the simulation and the real environment. In contrast, there is an important difference between queue time in both environments, which can be explained because the simulator is able to schedule a job without delay when there are available resources to execute a job. However, Condor middleware suffers for several delays due to different reasons such as delay notifications between distributed components, scheduling cycle duration or status update. To fix this error and reduce its influence on the results, two techniques are proposed: the first one adds a synthetic delay to the job execution time, whereas the second one adds the synthetic delay to the job queue time results. Also, how this feature can be incorporated in the simulator to get more accurate simulations is being studied for the meantime.

4 EXPERIENCE REUSE FOR THE SIMULATION OF A GLITE GRID

In this section, how a simulator for a multi-cluster Grid can be easily implemented replacing some parts of the previously developed simulator is shown. Also, we illustrate the usefulness of the methodology presented to validate the simulator results.

4.1 Overview of the AraGrid Grid

AraGrid[4] is a research and production Grid hosted by the Institute for Biocomputation and Physics of Complex Systems (BIFI)[5] and it is part of the European Grid Initiative (EGI)[6]. AraGrid consists of four homogeneous sites located at four different faculties in different geolocated cities. Every site is formed by 36 nodes with two 2.67 GHz 6-core Intel Xeon X5650 CPUs and 24 GB of RAM per node, making a total amount of 1728 cores and 4 TB of RAM. Both sites and nodes are interconnected by Gigabit links.

The Grid is managed by the gLite[7] middleware version 3.2.0 and every site use openPBS[8] version 2.5 as local batch system.

The AraGrid infrastructure is oriented to long-term experiment in the fields of physics, biochemistry, social behaviour analysis, astronomy, etc. Users are more conscious of loads and resource usage, although they deploy experiments similarly to the HERMES case, getting long waiting times.

4.2 Implementation and Validation of the AraGrid Simulator

Starting from the simulator structure, the design and implementation of the Condor simulator has been reused to develop a gLite simulator valid for the AraGrid computing infrastructure. This is an easy and quick implementation process, and the resulting simulator can be easily adapted to another gLite infrastructure. The reasons to implement these two simulators is twofold. On the one hand, HERMES (managed using Condor), AraGrid (gLite) and also PireGrid (gLite) are connected using high speed Gigabit links, which enhances data movement performance (which is left out of the scope of this paper). On the other hand, Condor and gLite are well known and

[4]http://www.araGrid.es/
[5]http://bifi.es/es/
[6]http://www.egi.eu/
[7]http://glite.cern.ch/
[8]http://www.mcs.anl.gov/research/projects/openpbs/

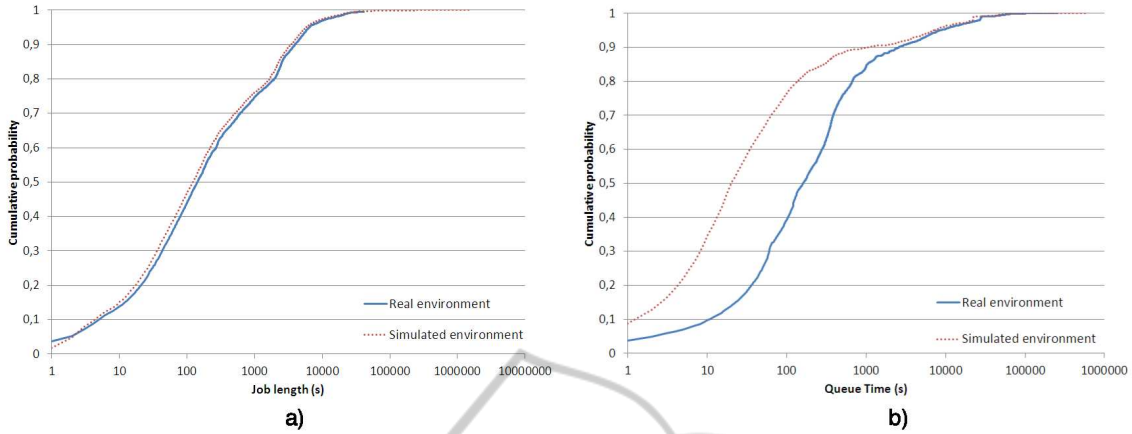

Figure 5: Job performance comparison between real data and simulation results in terms of: (a) job execution time, (b) job queue time.

widespread cluster/Grid middlewares in the research community.

The only component that needed a custom adaptation to fit the behaviour of AraGrid with respect to the HERMES simulator component is the scheduler. The scheduler's policy follows a hierarchical approach, as shown in Figure 6. Jobs sent by the Job Loader are managed by the global scheduler component that sends them to the right local scheduler considering job requirements, job rank and site occupation are taken. Meanwhile, every local scheduler uses a custom First Come First Serve (FCFS) policy.

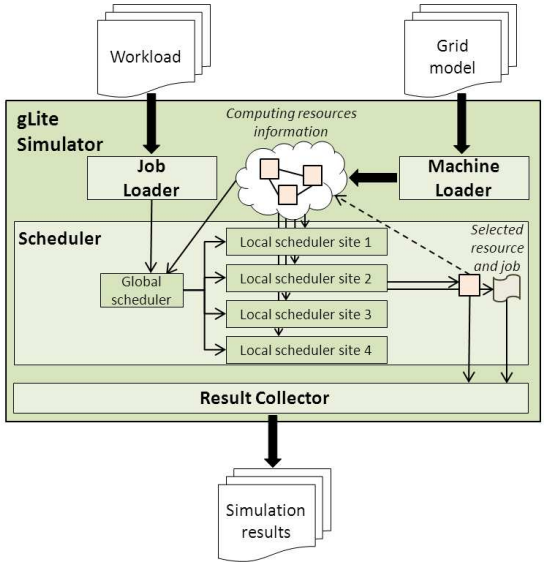

Figure 6: Architecture of the gLite simulator and detail of the local scheduler of a site.

It is important to consider a special case. As some sites are shared with other Grid initiatives such as EGI, the workload used as input contains jobs that can only be executed in shared sites. Sites where a job can be executed depends on the Virtual Organization (VO). Since this information is included in the workload, this special case can be properly treated by the scheduler when this kind of job reaches the global scheduler.

The resulting simulator component has been integrated into the AraGrid gLite mediator. The validation of the component has been carried out following the same approach depicted in subsection 3.3. In this case, the results are more accurate than in the HERMES case. That is because AraGrid scheduling policy is easier to replicate. The average error is of 1.19% with a standard deviation of 0.85%.

5 A CASE STUDY: INSPIRAL ANALYSIS WORKFLOW

In this section, the proposed simulation-based approach is applied in order to improve the performance of the Inspiral analysis scientific workflow. The experiment setup is detailed, with particular attention to the workload creation method used for modelling other users jobs that are executed in HERMES and AraGrid at the same time. Finally, performance results showing the benefits of our infrastructure are presented and discussed.

One of the main research lines of the Laser Interferometer Gravitational Wave Observatory (LIGO) is the detection of gravitational waves produced by various events in the universe (based on Einstein's theory of general relativity). The LIGO Inspiral Analysis Workflow is a scientific workflow which analyzes and tries to detect gravitational waves produced by various events in the universe using data obtained from the coalescing of compact binary systems such

as binary neutron stars and black holes (Taylor et al., 2006). Figure 7 depicts a simplified view of the main structure of the workflow. Although the workflow has a simple structure, it allows a high level of parallelism. As shown, the whole experiment is split into several smaller stages or blocks for analysis. The time-frequency data from any event for each of the LIGO detectors is arranged into template banks and used as an input for the workflow, which generates a subset of waveforms belonging to the parameter space and computes the matched filter output in each stage. Inspiral jobs are the most computationally intensive tasks in the workflow, generating most of the computing requirements. In case a true inspiral is detected, the matched filter output is computed and a trigger is generated and tested for consistency by the Thinca jobs as a result from the experiment. Finally, template banks are then generated from these trigger outputs and the process repeats.

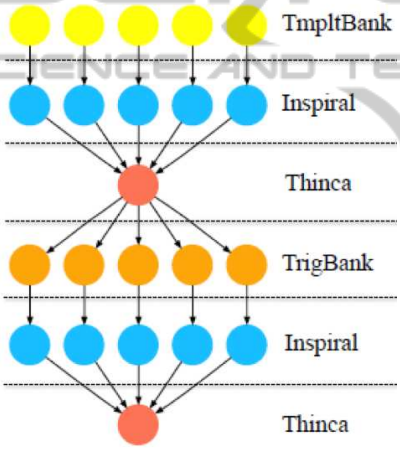

Figure 7: Workflow of the LIGO Inspiral analysis scientific workflow.

Several scientific workflows management systems could be used to develop the workflow. In our case, a high level Petri nets implementation (Reference nets) has been developed using the workflow editor provided by the framework depicted in Section 2. However, the workflow implementation details are out of the scope of this paper.

5.1 Experiment Setup

The experiment setup is not specific for this experiment or case study, but it is a general setup automatically generated by the components of the framework. This design simplifies the use of the infrastructure, making the simulation-based meta-scheduling completely transparent to the user.

The process is as follows: first, when a mediator retrieves a simulation request, it builds a workload describing the tasks to be simulated. Next, it gets information about the state of the computing infrastructure it represents. These data are used to adapt the predefined Grid model to its current situation (introducing resource failures) and to build a second workload representing the infrastructure state during the simulation. Details about the creation of this second workload are shown below. Once both workloads have been created, they are combined into one that is used as the simulation input. Then, the simulation starts its execution. Once it has finished, the simulation results are analysed by the mediator and only the information concerning the target tasks is provided to the meta-scheduler. Finally, the meta-scheduler chooses the best computing infrastructure based on data obtained from several simulations. For that purpose, the meta-scheduling policy uses a better-execution-time algorithm. Nevertheless, more complex policies involving the information obtained in previous simulations could be easily used.

The creation of the workload used to represent the state of the computing infrastructure is a key step in the simulation process. The importance of using an appropriate workload has been identified as a crucial input in some previous work (Feitelson, 2002; Li et al., 2004). Using a wrong workload can cause the simulation results not to correspond to the actual behaviour of the involved Grids. These research papers propose the generation of a single workload based on historical information from a long period of time and only considering representative periods (e.g. the peak hours during weekdays in job-intensive months). It is assumed that the longer the observation period is, the more representative is the workload, which allows tunning the Grid in extreme situations (Feitelson, 2002). Nevertheless, for simulation purposes these approaches are not valid because the state of the resources must be considered as the simulation starts. If an average or extreme workload is used, it is very likely to get very inaccurate results that lead to wrong scheduling decisions. Our proposal is to build several representative workloads with different situations depending on the state of the infrastructure (e.g. low load, average load and high load) and date. Therefore, the current computing infrastructure state is obtained before starting a simulation and used to select the most suitable workload. Also, the recovered infrastructure information, including currently running jobs and queued jobs, is added at the beginning of the workload, obtaining this way a workload describing the current infrastructure state and its evolution.

The model proposed in (Iosup et al., 2008b) has been used for workload creation. This model incorpo-

rates the notions of different users and jobs in *Bag-of-Tasks* (BoTs) to the Lublin-Feitelson model (Lublin and Feitelson, 2003). Due to the fact that the HERMES and AraGrid analysis has shown that more than 90% of jobs belongs a BoT and a few users are responsible for the entire load, this model is suitable for modelling jobs in our infrastructures.

5.2 Analysis of the Results

To prove the usefulness of the proposed approach, the workflow has been executed for a whole day (24 hours). Figure 8 depicts the CPU load observed in HERMES and AraGrid during the experiment. Note that HERMES load is different from the one sketched in figure 4. That is because the load in Figure 4 is an average load extracted from the execution log corresponding to the whole last year, whereas Figure 8 shows the cluster load on a particular day. As it can be observed, both computing infrastructures have different load models. Throughout the day there are better periods of time for submitting jobs to HERMES (mostly at early morning and night), and times more appropriate to submit jobs to AraGrid (in the afternoon). However, this is not the only criterion to be considered as the performance of a Grid infrastructure depends on many factors.

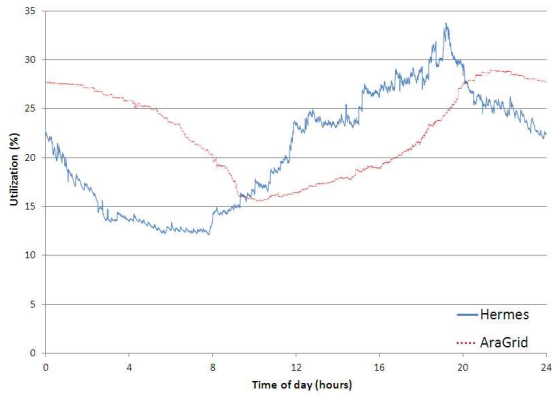

Figure 8: HERMES and AraGrid utilization (in percentage) observed during workflow execution.

The use of the simulation as a decision tool for meta-scheduling deals with this complexity and improves the performance obtained in the execution of the workflow as shown in Figure 9. The figure shows the total execution time for each stage of the Inspiral workflow entirely executed in each computing infrastructure (HERMES on the left bar and AraGrid on the right bar) and using the framework with the simulation-based meta-scheduling strategy (center bar) depicted previously. The results show that the use of the proposed approach leads to an improvement of 59% in HERMES execution time and a 111% in AraGrid execution time.

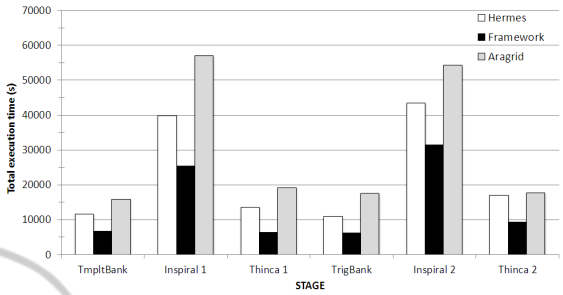

Figure 9: Experimental results for LIGO Inspiral analysis workflow.

Regarding the simulation overhead in terms of execution time, the simulation process for HERMES is more complex (more iterative structures) and can take up to 3-4 minutes for a bag of 10000 tasks, whereas for gLite it takes one minute approximately. Therefore, simulation times are insignificant in comparison to the execution time of each stage. Also, data movement has been measured. For the sake of clarity, as HERMES and AraGrid are connected by a Gigabit link, these times are small and can be avoided in the calculation of the overall execution time.

6 CONCLUSIONS

In this paper, a simulation component based on realistic workload usage has been presented. This component allows modelling and simulating different computing infrastructures in terms of performance, resource usage, cost, etc. We have also described a framework developed for the flexible deployment of scientific workflows in Grid environments, and which allows researchers to transparently work simultaneously with different and heterogeneous Grid middlewares.

The integration of the simulation component into the framework allows improving the meta-scheduling process. Not only a simulation process can be carried out to find the best computing infrastructure to execute a task (or a bag of tasks) in terms of performance or costs, but also the process may vary depending on the used workload. The use of realistic workloads provides very suitable and reliable results.

The flexible design and implementation of the simulation component also allows an easy adaptation for being used with different computing infrastructures, as it was shown by means of the reuse of the

HERMES simulator component (Condor) to develop the AraGrid one (gLite). Both Condor and gLite are two of the most used cluster/Grid middlewares in the research community. Thus, an additional advantage is that the developed components can be easily reused for simulating other existing computing infrastructures.

Finally, the integration of the presented approach into the framework has been applied to the development and execution of the Inspiral analysis over two different computing infrastructures, HERMES and AraGrid. As a result, the overall execution cost was significantly reduced.

Currently, the proposed simulation component is being extended to support the dynamic building of workloads. The use of dynamic workloads will minimize the effort required to build a new simulator and allow to obtain more accurate simulations. Also, the addition of new features in the simulator is being addressed in order to get more accurate queue times in simulations. Finally, the incorporation of complex meta-scheduling approaches that can use the information provided by the simulation process will be studied.

ACKNOWLEDGEMENTS

This work has been supported by the research project TIN2010-17905, granted by the Spanish Ministry of Science and Innovation.

REFERENCES

Abraham, A., Liu, H., Zhang, W., and Chang, T.-G. (2006). Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm. In *Proceedings of the 10th International Conference in Knowledge-Based Intelligent Information and Engineering Systems – KES 2006*, volume 4252, pages 500–507.

Fabra, J., Hernández, S., Álvarez, P., and Ezpeleta, J. (2012). A framework for the flexible deployment of scientific workflows in grid environments. In *Proceedings of the Third International Conference on Cloud Computing, GRIDs, and Virtualizations – CLOUD COMPUTING 2012*.

Feitelson, D. G. (2002). Workload Modeling for Performance Evaluation. In *Proceedings of Performance Evaluation of Complex Systems: Techniques and Tools – Performance 2002*, volume 2459, pages 114–141.

Foster, I. and Kesselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Hamscher, V., Schwiegelshohn, U., Streit, A., and Yahyapour, R. (2000). Evaluation of Job-Scheduling Strategies for Grid Computing. In *Proceedings of the First IEEE/ACM International Workshop on Grid Computing – GRID 2000*, pages 191–202.

Iosup, A. and Epema, D. H. J. (2011). Grid Computing Workloads. *IEEE Internet Computing*, 15(2):19–26.

Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., and Epema, D. H. J. (2008a). The Grid Workloads Archive. *Future Genereration Computer Systems*, 24(7):672–686.

Iosup, A., Sonmez, O., Anoep, S., and Epema, D. (2008b). The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th international symposium on High performance distributed computing – HPDC 2008*, pages 97–108.

Kertész, A. and Kacsuk, P. (2010). GMBS: A new middleware service for making grids interoperable. *Future Generation Computer Systems*, 26(4):542–553.

Klusáček, D. and Rudová, H. (2010). Alea 2 – Job Scheduling Simulator. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques – SIMUTools 2010*.

Li, H., Groep, D., and Wolters, L. (2004). Workload characteristics of a multi-cluster supercomputer. In *Proceedings of the 10th International Conference on Job Scheduling Strategies for Parallel Processing – JSSPP 2004*, pages 176–193.

Lublin, U. and Feitelson, D. G. (2003). The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122.

Ludwig, S. A. and Moallem, A. (2011). Swarm Intelligence Approaches for Grid Load Balancing. *Journal of Grid Computing*, 9(3):279–301.

Rahman, M., Ranjan, R., Buyya, R., and Benatallah, B. (2011). A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurrency and Computation: Practice and Experience*, 23(16):1990–2019.

Sargent, R. G. (2010). Verification and validation of simulation models. In *Proceedings of the 2010 Winter Simulation Conference – WSC 2010*, pages 166–183.

Sulistio, A., Cibej, U., Venugopal, S., Robic, B., and Buyya, R. (2008). A toolkit for modelling and simulating data Grids: an extension to GridSim. *Concurrency and Computation: Practice and Experience*, 20(13):1591–1609.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M. (2006). *Workflows for e-Science: Scientific Workflows for Grids*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Yu, J. and Buyya, R. (2005). A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record*, 34(3):44–49.

Yu, Z. and Shi, W. (2007). An Adaptive Rescheduling Strategy for Grid Workflow Applications. In *IEEE International Parallel and Distributed Processing Symposium, 2007 – IPDPS 2007*, pages 1–8.