DGSim: Comparing Grid Resource Management Architectures Through Trace-Based Simulation

Alexandru Iosup[⋆], Ozan Sonmez, and Dick Epema

Delft University of Technology, Delft, The Netherlands
`A.Iosup@gmail.com,{O.O.Sonmez,D.H.J.Epema}@tudelft.nl`
`http://www.pds.ewi.tudelft.nl/~iosup/dgsim.php`

Abstract. Many advances in grid resource management are still required to realize the grid computing vision of the integration of a world-wide computing infrastructure for scientific use. The pressure for advances is increased by the fast evolution of single, large clusters, which are the primary technological alternative to grids. However, advances in grid resource management cannot be achieved without an appropriate toolbox, of which simulation environments form an essential part. The current grid simulation environments still lack important workload and system modeling features, and research productivity features such as automated experiment setup and management. In this paper we address these issues through the design and a reference implementation of DGSim, a framework for simulating grid resource management architectures. DGSim introduces the concepts of grid evolution and of job selection policy, and extends towards realism the current body of knowledge on grid inter-operation, on grid dynamics, and on workload modeling. We also show through two real use cases how DGSim can be used to compare grid resource management architectures.

Key words: grid computing, simulation framework, trace-based simulation, performance evaluation

1 Introduction

Grid computing has emerged in the past decade as a middleware solution to integrating a world-wide computing infrastructure for scientific use. However, for the grid potential to become a reality, many of today's grid resource management (GRM) components such as job scheduling architectures, data management algorithms, and inter-operation mechanisms, need to be improved or even (re-)designed from scratch. In this process, simulation is critical to assess large numbers of experimental settings in a reasonable amount of time, and to design and evaluate the systems of the future. Despite steady progress, today's grid

[⋆] This work was carried out in the context of the Virtual Laboratory for e-Science project (`www.vl-e.nl`), which is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W), and which is part of the ICT innovation program of the Dutch Ministry of Economic Affairs. We would like to thank Shanny Anoep for his help with the development of the simulator.

simulators [1–7] still lack important features in system modeling, experiment setup, and experiment management. To address these issues, we present in this work DGSim, a framework for simulating GRM architectures.

Consider a researcher who wants to test the hypothesis "from two GRM architectures, architecture A is better than architecture B" using simulation. To this end, the researcher will have to perform several *experiments*, that is, simulations that cover (i.e., provide insights and deeper knowledge about) a family of real-world phenomena. One of the experiments is to assess the performance of A and B under realistic load. For this experiment, several *scenarios*, that is, groups of simulations that cover a single phenomenon, must be evaluated. One of the experiments can be to assess the performance of A and B under a realistic load of 50% of the system capacity. For each scenario, several *individual simulations* must be executed. A simulation is a computer program that represents or emulates the behavior of another system over time [8]. Finally, several *runs* are required for the same individual simulation to ensure statistical confidence in the results.

Our belief is that three decades of evolution in general computer simulation techniques are sufficient to enable the development and even optimization of grid resource management simulators (GRMS) by most interested scientists. In particular, many books [8–10] and survey articles [11, 12] attest to the progress made in parallel and distributed discrete event simulation. Therefore, we focus with DGSim less on generic simulation techniques, and more on the specific requirements of the GRM simulation process, for instance, by automatically generating realistic grid systems and workloads, by supporting typical GRM components, and by automating and speeding up the typical experiment (see Section 2). Our contribution is threefold:

– We design the DGSim framework for simulating GRM architectures, focusing on workload and system modeling, and on automatic experiment setup and management ("from configuration file to results");
– We make methodological improvements to the area of GRMS, by introducing the concepts of grid evolution and job selection policy, and by extending the state-of-the-art for grid inter-operation, grid dynamics, and workload modeling;
– We present through two real use cases how DGSim can be used in practice.

2 Requirements for Simulating Grid Resource Management Architectures

In this section we synthesize the design goals for completing the path from hypothesis to simulation results in grid resource management research. Our motivation is twofold. First, this type of research has specific requirements, e.g., unique experimental setup characteristics. Note that these requirements do not only fit grids, but also large-scale distributed computing systems in general. Second, in spite of the evolution of GRMSs over the last decade (see Section 5), there is still place for improvement, especially in automating the experimental setup and in supporting large-scale experiments. Below we present seven requirements, divided into three categories, specific to simulation frameworks for GRM research; in Section 5 we use them to evaluate the related work.

1. **Experimental Setup** A major challenge in today's grid simulation landscape is designing a realistic experimental setup. Specifically:
   (a) **Grid System** There is a need to generate models of grids that include the topology, resource, and inter- and intra-operation (e.g., service level agreements) of real grids. The model may also include varying resource availability.
   (b) **Grid Dynamics and Evolution** It is important to evaluate resource management solutions for future system configurations. With grid systems being highly dynamic in the number and the size of their resources on both short and long term (see also Section 3.3), there is need to generate grid dynamics and evolution events.
   (c) **Grid Workload** To ensure that tests are performed under conditions that match the test design, there is a need to synthesize realistic grid workloads.
2. **Experiment Support for GRMs** Perhaps the most limiting factor in the adoption of a simulation framework is the degree of automation.
   (a) **Performing Grouped Simulations** There is a need to automate the execution of groups of individual simulations, e.g., for a specific scenario. The simulation frameworks should be able to run in an unreliable environment, i.e., simulating grids *on* grids.
   (b) **Performing Batches of Simulations** There is a need to automate the execution of batches of simulations with different parameter settings. The simulation frameworks must generate the corresponding configuration files, ensure that the simulator is executed, and collect the results for on-line and off-line analysis.
   (c) **Storing and Analyzing Simulation Results** Large amounts of data are produced when simulating grid resource management solutions. As a first step towards Online Analytical Processing (OLAP) capabilities, there must exist a hierarchy for results storage and aggregation, i.e., by experiment, by scenario, by run.Second, to promote the comparability of results across different research groups, a standard procedure for analysis and presentation of results must be followed. In particular, the most common grid resource management metrics should be automatically extracted from the simulation results.
3. **Performance** Given the sizes of today's clusters, and even those of real experimental grids, there is a need for simulated environments of at least several thousands of processors. At the same time, given the high percentage of single processor jobs present in grid workloads [13], there is a need for a simulator to deal with workloads of tens of thousands to even millions of jobs. We therefore describe the performance of a simulation framework with five metrics: (M1) experimental setup time, (M2) the number of scenarios, (M3) the number of individual simulations, (M4) the size of the simulated systems, and (M5) the workload to which the simulated systems are subjected.

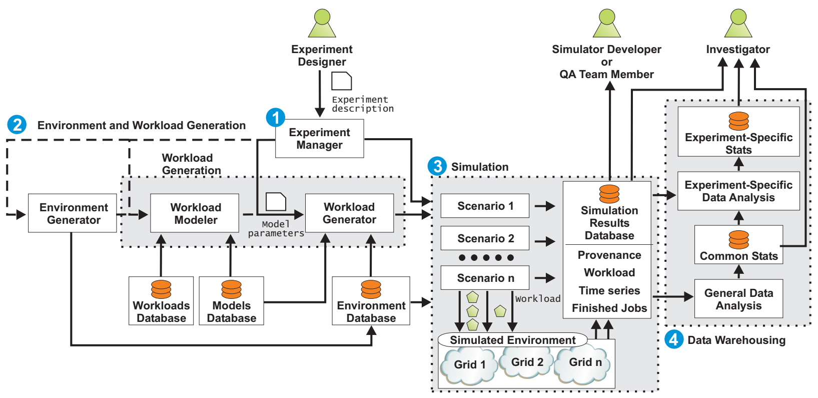

Fig. 1. An overview of the design of the DGSim Framework.

The individual simulation time is not among these metrics, as it depends heavily on what is actually simulated, e.g., the execution of a simulator of an NP-complete scheduling algorithm may not finish regardless of the actual capability of the simulation engine. Similar to the discussion about workload realism, good performance depends on the specifics of the experiment.

3 DGSim: A Framework for Simulating Grid Resource Management Architectures

In this section we discuss the design of DGSim.

3.1 Design Overview

We envision four main roles for the DGSim user. The *Investigator* formulates the problem and is interested in the results. The *Experiment Designer* is responsible for designing and performing a set of experiments that give answers to the problems raised by the Investigator. The *Simulator Developer* helps the Experiment Designer perform the experiments by implementing novel components for the simulator, e.g., a new scheduling algorithm, or a new analysis tool. The *QA Team Member* is responsible for validating the experiments, e.g., by duplicating simulation results.

Figure 1 shows an overview of the DGSim design. The arrows represent the direction of data flows from components. The main components of DGSim are:

1. The Experiment Manager coordinates the flow of an experiment. If the Experiment Designer did not fix the experimental environment, it calls the Environment Generator. Similarly, if the workload model parameters are not fixed, it calls the Workload Modeler. After generating the environment and the workload, the Experiment Manager calls the Simulation component.
2. The Environment and Workload Generation component automates the experimental setup. The Environment Generator will generate for each environment the starting topology, which includes the resource specification and the logical inter-connection between resources, and the system dynamics and evolution (see Section 3.3). Similarly, the Workload Generation ensures that realistic workloads are generated or simply extracted from existing grid

workload traces. This component can also automatically extract parameters for a given model from an input grid workload trace.

3. The Simulation component executes large numbers of individual simulations grouped by scenario and run. The computing power may be provided by one machine (e.g., the scientists' computer) or by a grid. The Simulation component also collects all the results and stores them into the Simulation Results Database for future analysis. The stored data are the simulation results and provenance data (simulator parameters, version, and events).
4. The Data Warehousing component is responsible for organizing data, and for data analysis. Data are indexed by experiment, by scenario, and by run. This component analyzes for all simulations the time series of job arrivals, starts, and completions, and produces statistical results for the common performance metrics, e.g., time-related, related to resource consumption, and related to resource waste. It also analyzes the messages exchanged by the simulated entities. Additional analyzers may be used to process the data.

3.2 A Model for Inter-Operated Cluster-Based Grids

In this section we present the system model used in the DGSim framework. Our model extends previous grid simulation approaches (see Section 5) in several ways. In previous work, all the jobs in a queue are considered as the input set of the scheduler; our job scheduling model also allows the selection of only some of the queued jobs, through a selection policy. In current grid simulators, the resource management architecture is either purely centralized or purely decentralized. In comparison, the inter-operation model of DGSim also considers the hierarchical and the (more realistic) hybrid architectures. Finally, existing simulators have used an information model in which only one of the job runtime, the cluster dynamic status, and the resource speed is not (accurately) known. In contrast, the information model of DGSim allows all the pieces of information used in the scheduling process to be inaccurate or even missing.

The resource model assumes that grids are groups of clusters of homogeneous resources, e.g., processors, network, storage. There is no logical connection between them in the simulator; the simulator ensures when resources are used simultaneously or in some dependency. The clusters may have logical links with other clusters (e.g., see Section 3.2); the clusters may be virtual, that is, contain no resources. We use the Worldwide LHC Computing Grid (WLCG) [14] as a running example. The computational speed of a resource is modeled similarly to WLCG's[1], that is, in SPECInt2K[2]; the values for WLCG's processors speeds range from 0.1 to 6.0, with an average of 1.184.

The workload model takes into account individual jobs, job grouping (e.g., batches, workflows), and the user/VO to whom the job belongs. We also introduce the notions of a user's local cluster, where all the jobs of a user are sub-

[1] For all the WLCG-related information used in this paper, the data sources are `http://goc.grid.sinica.edu.tw/gstat/` and `http://pcalimonitor.cern.ch/reports/`.

[2] SPEC's Int2K, [Online] Available: `http://www.spec.org/cpu2000/`.

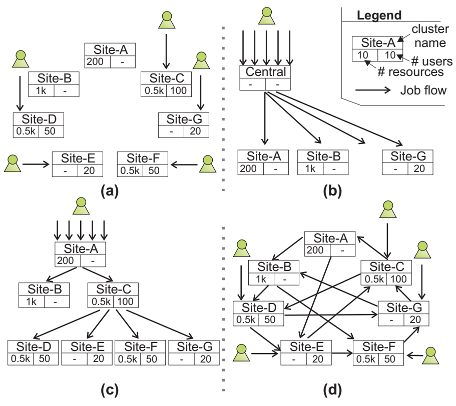

Fig. 2. Grid inter-operation: (a) independent clusters; (b) centralized meta-scheduler, (c) hierarchical meta-scheduler; (d) distributed meta-scheduler.

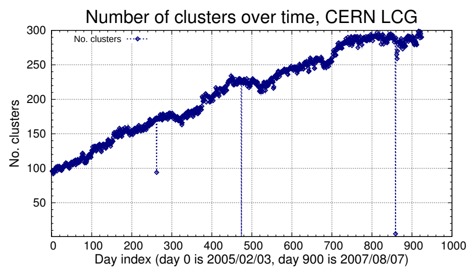

Fig. 3. The evolution of the number of clusters over time in WLCG.

mitted, and of job's arrival cluster. We assume the execution time of a job to be proportional to the speed of the processor it runs on.

The inter-operation model While many of today's grids still operate in isolation, an integration trend has recently emerged, similar to what happened in the past decades with the growth of the Internet from isolated networks. We therefore integrate in our simulation framework the concept of grid inter-operation. For this purpose, we have implemented six architectural alternatives, of which four are depicted in Figure 2. The independent (or isolated) clusters architecture assumes there is no load exchange between the clusters. The centralized meta-scheduler assumes the load exchange is coordinated by a central node. With hierarchical meta-scheduler, the load exchange is coordinated by a hierarchy of schedulers. The distributed meta-scheduler assumes there is no central control for load sharing and each cluster can exchange load with any other cluster. For the multi-level architectures, DGSIM already provides several load distribution policies, e.g., Worst- and Best-Fit, Round-Robin, and load-dependent.

The job scheduling model Each cluster contains one or more scheduling queues. From each queue, the queue's *job selection policy* extracts an eligible set of jobs to be scheduled; the eligible set can be ordered by tagging the extracted jobs. The eligible set is scheduled on the local resources using the queue's *job scheduling policy*. The events which trigger the job selection and the job scheduling policies are dependent on the simulated setup, that is, they can be set by the user. A DGSIM user can easily modify or re-implement the selection and the scheduling policies at various levels of the GRM architecture, using the Python scripting language. DGSIM already provides seven selection policies, and over ten scheduling policies, including FCFS, two FCFS/backfilling variants, and eight scheduling policies for bags-of-tasks [15]. While we have not experimented with

with several queues at the same cluster, e.g., to simulate a cluster with per-VO queues, the DGSim user can implement such a setup and use the workload model's user/VO characteristics to set the arrival queue of a job, and prioritizing the treatment of queues.

The information model deals with the accuracy and the timeliness of the information in an inter-operated (distributed) grid. A scheduling policy that deals with a job needs pieces of information that may be unavailable, inaccurate, or unknown in a realistic setting: the job's amount of work to be computed, a remote cluster's dynamic status, and even the remote cluster's static resource characteristics. Information inaccuracy can be the result of predictions; the DGSim implementation currently provides over ten predictors widely used in large-scale distributed systems [16].

3.3 Grid Dynamics and Grid Evolution

We have already argued that grids change over time. We identify two types of change: over the short term, determined by dynamic resource availability (e.g., processors failing), and over the long term, determined by static resource availability (e.g., the addition of a new cluster to a grid, or of new processors to its clusters). We call the former *grid dynamics*, and the latter *grid evolution*. Only one of the surveyed grid simulators considers grid dynamics, i.e., GSSIM [6]; none of them considers grid evolution. Moreover, in comparison with GSSIM, our grid dynamics model also considers the concept of correlated failures, that is, of failures that affect several resources at the same time; this phenomenon is common in and important for the performance of both clusters and cluster-based grids [17, 18].

The grid dynamics model considered by DGSim describes the changes in resource availability status, and includes the following aspects: the clusters where a change occurs, the number of resources involved in the change, the time when the change occurs, and the time until the next status change for the same resource(s). DGSim currently implements the realistic grid dynamics model proposed in [18].

The grid evolution model The grid evolution phenomenon is depicted in Figure 3: in less than three years, the WLCG has grown from 100 to 300 clusters; during the same period, the median cluster size has increased by just 20% (not shown). To account for grid evolution, we have designed a generator that for a grid system outputs the topology and resources for a time frame that spans from "present" to "far-future". A researcher can opt to perform simulations only for the present grid system, or for several sizes up to the far-future.

3.4 Grid Workload Generator

One of the most important problems when performing simulations of computer systems is setting the input workload. The experiment results may not be meaningful when using unrealistic workload models. To assess the behavior of a resource management component, workloads that incur the same average system load but have very different characteristics need to be generated. Moreover, it is sometimes useful to have some overlap of the characteristics of the generated input, to put in evidence the impact of the non-overlapping characteristics; then, only the values of some of the model parameters can be changed.

Realistic workload models DGSim currently supports two realistic workload models: the Lublin-Feitelson (LF) model for jobs in parallel supercomputers and clusters [19] and a new model for bags-of-tasks in grids; a detailed description of the latter is beyond the scope of this article. The LF model considers for a job its parallelism, its runtime, and their correlation, and for the whole workload the arrival patterns (i.e., daily cycle, peak hours); this model has been validated using four long-term traces from parallel production environments and used by numerous grid researchers [20, 21]. The model for bags-of-tasks in grids considers, in addition to the LF model, the individual users and the grouping of jobs in bags-of-tasks; this model has been validated using seven long-term grid traces, including five from the Grid Workloads Archive (GWA) [22].

Iterative workload generation The realistic grid workload models are complex, that is, they have many parameters and correlations between parameters. Generating a workload with a desired load from a complex model is difficult, especially when only some of the model parameters can be changed. To this end, we have developed an Iterative grid Workload Generation algorithm (IWG). The IWG algorithm takes as input a description of the target system (S), the target load (L), the maximum deviation from the target load ϵ, the workload model, and the fixed parameters of the model. It outputs a workload generated with the model such that the workload incurs on the target system the target load (taking into account ϵ). To achieve this, the algorithm tries to iteratively change (increase or decrease) the value of a non-fixed parameter, e.g., the job inter-arrival rate. The algorithm quits after a fixed number of iterations.

4 Experiments using DGSim

We have already used DGSim in a variety of experiments. In this section we present a summary of these experiments, with the purpose of demonstrating the usefulness of our framework. For each experiment we describe the problem and the setup; for more details we refer to [18, 21].

4.1 Performance Evaluation Using Real Workload Traces

Summary Using DGSim, we have assessed the behavior of five grid resource management architectures under real load [21].

Setup For this experiment, we have simulated the inter-operation of the DAS-2 and Grid'5000 grids, so that together they constitute a system with 20 clusters and over 3000 resources. The input workload was a contiguous subset of 12 months from the real grid traces taken from DAS-2 and Grid'5000, starting from 01/11/2005 [22]. Figures 4 and 5 show the configuration files for the environment and for the simulator setup, respectively. Figure 6 depicts a comparative display of the number of jobs started by each of the five architectures over time. At the middle of days 26 and 27 the number of jobs submitted to one of the clusters surges, and `DMM` and `fcondor` start migrating load across clusters (the

```
ID  Cluster       NProcs
c01 DAS/fs0       144
...
c06 G5K/site1/c1  128
c07 G5K/site1/c2  128
...
```

Fig. 4. The configuration file for the environment setup.

```
Sim     PlugIn
cern    sim_cern.py
condor  sim_condor.py
DMM     sim_dmm.py
fcondor sim_fcondor.py
koala   sim_koala.py
```

Fig. 5. The configuration file for the simulator setup.

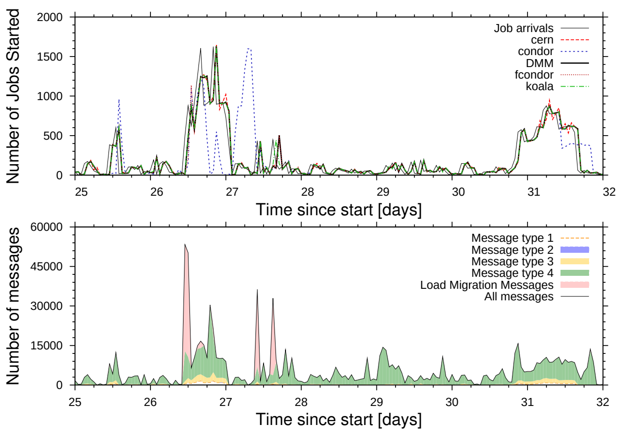

Fig. 6. System operation over time: (*top*) comparison of the number of jobs started by five grid resource management architectures. (*bottom*) the number of messages for DMM, per message type.

```
experiment.ID = E2-1D
scenario.SimTypes = DMM,sep-c,cern,condor,fcondor,koala
scenario.Loads = 10,30,50,60,70,80,90,95,98
scenario.TracesBaseDir = K:\EuroPar08\traces\10x\
scenario.BaseOutputDir = K:\EuroPar08\results\
...
```

Fig. 7. The configuration file for the second experiment.

bottom row depicts the messages of DMM). This demonstrates how using DGSim's automated analysis features, including graphing, facilitates understanding the causes of the performance variations across various grid configurations.

4.2 Performance Evaluation Using Realistic Workload Traces

Summary Using DGSim, we have assessed the performance of six grid resource management architectures under realistic load from 10% to 98% [21].

Setup Similarly to the previous experiment, we have simulated the inter-operation of the DAS-2 and Grid'5000 grids. First, the workloads are generated using the LF model with the parameter values extracted automatically from the grid traces present in the GWA [22]. Then, the experiment is run automatically; in particular, all the configuration files (540, that is, 6 architectures $\times$ 9 loads $\times$ 10 repetitions of each simulation for statistical soundness) are automatically generated. Finally, the results (over 20GB of data) are automatically analyzed to produce more than 20 common performance metrics. Throughout the process the tools use the same experiment configuration file, depicted in Figure 7. This configuration file specifies, among others, the experiment unique identifier (tag `experiment.ID`), the six simulated architectures (tag `scenario.SimTypes`), the nine loads under which the simulated system will operate (tag `scenario.Loads`), and the input and output base directories (tags

Table 1. A summary of simulators in grid computing research. The metrics M1-5 are evaluated for the largest reported set of experiments. Acronyms: T – from trace, M – from model, D – dynamic, G – generate until reaching the user-specified goal.

		Experimental			Experiment			Performance					
	Distinctive	Setup			Support			M2	M3	M4		M5	
Simulator	Feature	a	b	c	a	b	c	scen.	sim.	proc.	sites	jobs	load
BeoSim [3]	co-allocation	-	-	M	-	+	+	350	20k	100	5	4M	-
ChicSim [2]	data replication	-	-	M	-	-	-	12	72	-	30	6k	-
CSim/Grid [23]	co-allocation	-	-	M	-	+	-	100	4k	200	4	-	70%
GangSim [5]	usage SLAs	-	-	T	-	+	+	10	100	300	20	1k	-
GridSim [1, 24, 25]	economic SLAs	-	-	TM	-	-	-	100	100	10k	15	5k	-
GSSIM [6]	automation	TMD	M	M	-	-	-	1	1	4	1	10	-
MONARC [26]	CERN	-	-	M	-	-	+	100	100	2k	5	5k	-
OptorSim [4]	data replication	-	-	-	-	-	+	50	0	-	20	10k	-
SimGrid [15, 27, 7]	batch&DAG jobs	TMD	-	TM	-	+	-	100	10k	300	-	4k	-
DGSim	see Section 3	TMD	TM	TMG	+	+	+	100	6k	3k	20	1.4M	300%

`scenario.TracesBaseDir` and `scenario.BaseOutputDir`, respectively; the last part of the `scenario.TracesBaseDir` tag (`10x`) shows that there are 10 sets of traces in the input directory, one for each experiment repetition).

5 Related work

In this section we survey several approaches to simulation of grid computing systems. We assess the relative merits of the surveyed approaches according to the requirements described in Section 2. Table 1 summarizes our survey; other simulators we review here [28–30, 20, 31] may improve marginally on this body of knowledge. All the surveyed simulators have a discrete model representation, that is, the simulation model changes state only at discrete points in simulation time[3]. With the exception of GangSim and MONARC, which are discrete time-stepped simulators, all the surveyed simulators are discrete event simulators. To speed-up the simulation, DGSim can run on clusters and grids to execute multiple individual simulations in parallel, while MONARC and SimGrid use multi-threading to speed-up individual simulations. Compared to the previous simulation tools, DGSim focuses more on the simulation process, with richer grid system and workload generation, and support for large numbers of individual and grouped simulations. This is particularly visible in the size of the reported experiments: DGSim can use workloads orders of magnitude larger than the other simulators (with the exception of BeoSim), on similarly sized simulated environments; BeoSim can work with a workload of similar size, but on a system that is an order of magnitude smaller.

6 Conclusion and future work

The evolution of today's grids into the technological solution of choice for sharing computing resources depends on the ability of its developers to improve significantly the current grid resource management systems. To accomplish this difficult task, a toolbox in which simulators play a critical role is needed. In this paper we have proposed DGSim, a framework for simulating grid resource management architectures, which focuses on automating and optimizing the overall

[3] For all the terms related to simulation we refer to the textbook of Fujimoto [8].

simulation process, from hypothesis to obtaining simulation results. From the methodological side, DGSim introduces or extends the concepts of grid inter-operation, grid resource dynamics, grid evolution, and grid workload model. We have presented in this work the design, the reference implementation, and two real cases where DGSim was used.

For the near future, we plan to continue the development of DGSim by extending it with libraries of algorithms and mechanisms for job scheduling, data management, and grid inter-operation. This extension will allow scientists to setup simulations in which these algorithms can be readily compared with alternatives.

References

1. Buyya, R., Murshed, M.M.: GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. C&C: Practice and Experience **14**(13-15) (2002) 1175–1220
2. Ranganathan, K., Foster, I.T.: Decoupling computation and data scheduling in distributed data-intensive applications. In: HPDC, IEEE CS (2002) 352–358
3. Jones, W.M., Pang, L.W., Stanzione Jr., D.C., Ligon III, W.B.: Job communication characterization and its impact on meta-scheduling co-allocated jobs in a mini-grid. In: IPDPS. (2004)
4. Cameron, D.G., Millar, A.P., Nicholson, C., Carvajal-Schiaffino, R., Stockinger, K., Zini, F.: Analysis of scheduling and replica optimisation strategies for data grids using OptorSim. J. Grid Comput. **2**(1) (2004) 57–69
5. Dumitrescu, C., Foster, I.T.: GangSim: a simulator for grid scheduling studies. In: CCGRID, IEEE CS (2005) 1151–1158
6. Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Grid scheduling simulations with GSSIM. In: SRMPDS. (2007)
7. Caron, E., Garonne, V., Tsaregorodtsev, A.: Definition, modelling and simulation of a grid computing scheduling system for high throughput computing. FGCS **23**(8) (2007) 968–976
8. Fujimoto, R.M.: Parallel and Distribution Simulation Systems. John Wiley & Sons, Inc., NY, USA (1999)
9. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. 3rd edn. McGraw Hill, NY, USA (2000)
10. Banks, J.S., Carson II, J., Nelson, B.L., Nicol, D.M.: Discrete-Event System Simulation. 3rd edn. Prentice-Hall, Inc., New Jersey, USA (2001)
11. Fishwick, P.A.: Simulation model design. In: WSC, ACM Press (1995) 209–211
12. Perumalla, K.S.: Parallel and distributed simulation: traditional techniques and recent advances. In: WSC, ACM Press (2006) 84–95
13. Iosup, A., Dumitrescu, C., Epema, D.H., Li, H., Wolters, L.: How are real grids used? The analysis of four grid traces and its implications. In: GRID, IEEE CS (2006) 262–270
14. : Worldwide LHC Grid Computing. [Online] `http://lcg.web.cern.ch/LCG/`. (2007)
15. Casanova, H., Legrand, A., Zagorodnov, D., Berman, F.: Heuristics for scheduling parameter sweep applications in grid environments. In: HCW. (2000) 349–363
16. Sahoo, R.K. et al.: Critical event prediction for proactive management in large-scale computer clusters. In: KDD, ACM Press (2003) 426–435

17. Zhang, Y., Squillante, M.S., Sivasubramaniam, A., Sahoo, R.K.: Performance implications of failures in large-scale cluster scheduling. In: JSSPP. Volume 3277 of LNCS., Springer-Verlag (2004) 233–252
18. Iosup, A., Jan, M., Sonmez, O., Epema, D.H.: On the dynamic resource availability in grids. In: GRID, IEEE CS (2007) 26–33
19. Lublin, U., Feitelson, D.G.: The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. PDC **63**(11) (2003) 1105–1122
20. Casanova, H.: On the harmfulness of redundant batch requests. In: HPDC, IEEE CS (2006) 70–79
21. Iosup, A., Epema, D., Tannenbaum, T., Farrellee, M., Livny, M.: Inter-operating grids through delegated matchmaking. In: SC, ACM Press (2007)
22. : The Grid Workloads Archive. [Online] `http://gwa.ewi.tudelft.nl`. (2007)
23. Bucur, A.I.D., Epema, D.H.J.: Trace-based simulations of processor co-allocation policies in multiclusters. In: HPDC, IEEE CS (2003) 70–79
24. Ranjan, R., Buyya, R., Harwood, A.: A case for cooperative and incentive-based coupling of distributed clusters. In: CLUSTER, IEEE CS (2005)
25. Sulistio, A., Poduval, G., Buyya, R., Tham, C.K.: On incorporating differentiated levels of network service into GridSim. FGCS **23**(4) (2007) 606–615
26. Legrand, I., Newman, H.B.: The MONARC toolset for simulating large network-distributed processing systems. In: WSC. (2000) 1794–1801
27. Legrand, A., Marchal, L., Casanova, H.: Scheduling distributed applications: the simgrid simulation framework. In: CCGRID, IEEE CS (2003) 138–145
28. Takefusa, A., Matsuoka, S., Nakada, H., Aida, K., Nagashima, U.: Overview of a performance evaluation system for global computing scheduling algorithms. In: HPDC, IEEE CS (1999)
29. Phatanapherom, S., Uthayopas, P., Kachitvichyanukul, V.: Fast simulation model for grid scheduling using HyperSim. In: WSC, ACM (2003) 1494–1500
30. He, L., Jarvis, S.A., Spooner, D.P., Bacigalupo, D.A., Tan, G., Nudd, G.R.: Mapping dag-based applications to multiclusters with background workload. In: CCGRID. (2005) 855–862
31. Ramakrishnan, A et al.: Scheduling data-intensive workflows onto storage-constrained distributed resources. In: CCGRID, IEEE CS (2007) 401–409