

Scheduler simulation using iSPD, an iconic-based computer grid simulator

Denison Menezes, Aleardo Manacero, Renata Spolon Lobato, Diogo T. da Silva
 Computer Science and Statistics Dept
 Univ. Estadual Paulista - UNESP
 Rio Preto, Brazil
 Email: denison.m3@gmail.com, aleardo@ibilce.unesp.br

Roberta Spolon
 Department of Computing
 Univ. Estadual Paulista - UNESP
 Bauru, Brazil

Abstract—Increased accessibility to high-performance computing resources has created a demand for user support through performance evaluation tools like the iSPD (iconic Simulator for Parallel and Distributed systems), a simulator based on iconic modelling for distributed environments such as computer grids. It was developed to make it easier for general users to create their grid models, including allocation and scheduling algorithms. This paper describes how schedulers are managed by iSPD and how users can easily adopt the scheduling policy that improves the system being simulated. A thorough description of iSPD is given, detailing its scheduler manager. Some comparisons between iSPD and Simgrid simulations, including runs of the simulated environment in a real cluster, are also presented.

Keywords—grid simulation; grid computing; scheduling policy;

I. INTRODUCTION

The demand for high-performance computing has risen sharply during the last decade. This trend is driven by the need for accuracy and to solve problems with large amounts of data. While parallel computing is the recognized approach to provide performance, its associated costs motivated developments toward clusters, grids and, more recently, computing clouds. Computer grids, specially, provided a very useful ground for several applications in fields where users are not experts in parallel computing or performance analysis. The main reason behind this is that the low cost of deploying a grid environment, which is mainly based on sharing existing resources and has large computing power [1], [2], [3], made grids available to anyone wishing to share their resources.

In distributed systems, such as computer grids, scheduling policies have a great impact on the system's performance. Unfortunately, choosing the correct policy for a given environment is not an easy task as it usually demands a big effort to evaluate the available policies in order to select the right one. This evaluation may be performed through simulation, avoiding the use of the real system for measurements [4]. Simulation is also more flexible and less expensive than pure benchmarking.

There are several grid simulators available, such as Simgrid [5], and Gridsim [6] and others. However, none of them provide an easy-to-use interface to model the system, demanding knowledge about scripting and/or programming. In order to circumvent these problems, iSPD (iconic Simulator of Parallel and Distributed systems) [7] uses an iconic-based approach to create system models, and includes a task scheduler manager that is presented here. While the modelling approach used in iSPD was firstly presented in [7], this paper is concerned with the analysis of grid schedulers using it.

In the following sections one finds a general description of iSPD, followed by a thorough description of its simulation engine, including the management of scheduling policies. After that, results from iSPD simulations are compared to results from Simgrid. Additional results comparing the execution of actual programs in a cluster and their simulation with iSPD are also provided. To conclude, a brief review of similar works is presented, followed by general conclusions about iSPD.

II. ICONIC SIMULATOR OF PARALLEL AND DISTRIBUTED SYSTEMS – iSPD

iSPD is a simulation framework developed and made available from the Parallel and Distributed Systems Laboratory at Paulista State University [8]. It provides an easy interface to create grid models that could be used by people that are not expert in programming scripts or other simulation languages. It is based on iconic modelling, creating models that can be translated to queue models before simulation [7]. The general architecture of iSPD is shown in Figure 1, where it can be seen that the user inserts a model in the interface, which generates a model in an iconic language. This model is converted to a model in a queuing language before going to the simulation engine to produce performance metrics for the user.

The reason for using two different languages during the simulation process is the separation between the simulator and the modelling interface. With this separation, it is very easy to create converters of models for different simulators

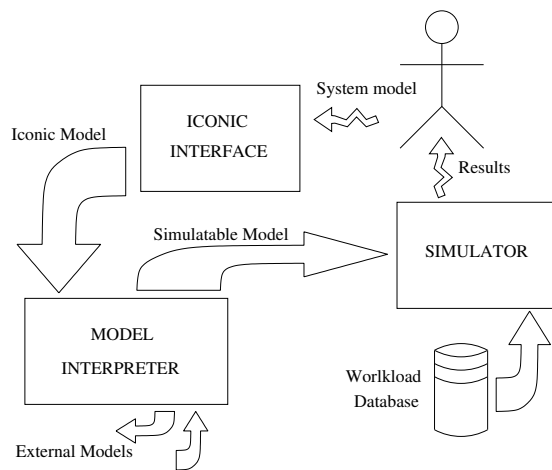


Fig. 1. ISPD's general architecture [7]

to iSPD models, and vice versa. This enables the reuse of models already created for other simulators into iSPD, even allowing the user to modify the original model through the iSPD interface.

At this time there are some restrictions to the systems that can be simulated. They are:

- Tasks have to follow the bag-of-tasks model, and have attributes for load, time of occurrence, and owner;
- Parallelism is modelled through the master-slave paradigm;
- Standard scheduling policies are round-robin, FIFO, Workqueue, WQR (WorkQueue with Replication) and Dyn-FPLTF (Dynamic Fastest Processor to Largest Task First);
- Standard performance metrics are turnaround-time, waiting-time, user satisfaction, and efficiency.

iSPD is implemented in Java and contains three major components, as seen in Figure 1. The iconic interface and the simulation engine interact with the model interpreter, which is responsible for converting iconic models to queue models. Each component is briefly described in the following.

A. Iconic interface

A GUI provides some basic functionality to model computer grids. The available icons represent hosts, clusters, one-way communication channels, and network cloud (internet). In this interface, the user can graphically model a system and provide all needed parameters (loads, processing and communication speeds, scheduling policies, etc.). It is an intuitive interface and allows for a wide range of distributed systems models.

B. Model interpreter

The model interpreter is the link between the iconic interface and the simulation engine. Basically, it is a multilingual interpreter whose target languages are either the queuing language, used in simulation, or the iconic language, used by the iconic interface. The former target comes when interpreting iconic models; the latter is originated from external ones. Currently, iSPD is capable of translating scripted models

written for Simgrid into iconic models. Other conversions are under work. It should be observed that this functionality was very useful to validate iSPD models against Simgrid models.

The translation process, in any direction, is defined by the respective grammars for the iconic, queuing and external models. These grammars are described in [7]. Since the description of such languages is outside the scope of this paper, it is enough to say that they are context-free grammars, and their analyzers were produced using this characteristic to enable easy and fast interpretation.

The existence of an interpreter for external models suggests the possibility of an opposite conversion, that is, a conversion from iconic models to language scripts for external grid simulators. This is one of the new features under development.

C. Simulation engine

This is the component that effectively conducts the simulation process. It reads the system's model after its conversion from the iconic to the queuing model. It is an event-based simulator with two basic modules. One that manages the service centres, including queues and their respective servers, and another that manages the scheduling policies used by each server. The possible events managed by the simulation engine are:

- **Task arrival:** when a specific task is added to a server's queue;
- **Task service:** when a task is served by the server;
- **Task delivery:** when a task is removed from a service centre and, eventually, creates a new **Task arrival** in another centre.

Figure 2 shows how the simulator works. It follows the Event Scheduling/Time Advance model [9], managing future events (FE) list through correct insertions and removals, and adjusting the simulated time. This means "executing" the most immediate event in the FE list until all events have been served. When no more events are present in the list, the engine produces all relevant metrics that are going to be used to evaluate the grid or any distributed system.

In the simulation engine, the computational infrastructure for a grid is mapped to a queuing network linking service centres. Each service centre represents specific operations (or icons from the modelling interface) for the system, such as communication centres and processing centres. Specific centres include:

- **Communication service centres**
 - **Direct link centres:** follow the one queue-one server model, with a FIFO service policy. These centres are used to connect two other centres in order to move data around the grid;
 - **Switching centres:** follow the multiple queues-one server model, with a FIFO service policy. They are used to connect cluster nodes;
 - **Internet center:** follows a one queue-multiple servers model, where the number of servers is allowed to grow indefinitely. This allows the emulation of a zero length queue that is fed by, and feeds, other communication service centres;

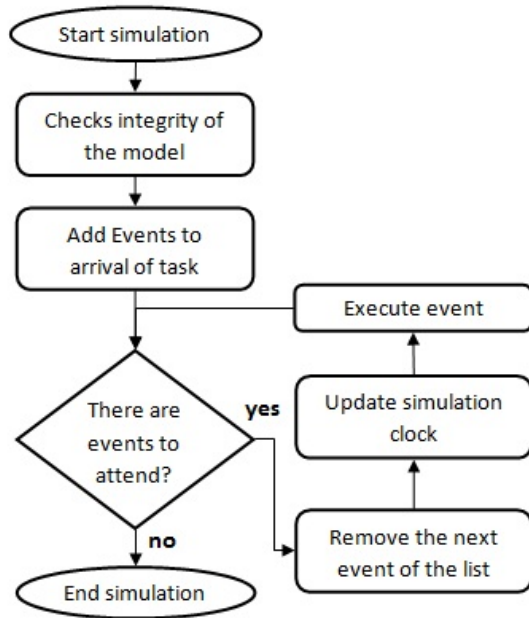


Fig. 2. Simulation Process

• Processing service centres

- **Host service centres:** follow a one queue-multiple servers model, with a FIFO policy. This type of center emulates the jobs processing, containing one or more servers to represent single- or multiprocessor hosts with shared memory;
- **Centralized server service centres:** follow a one queue-multiple servers model, with FIFO policy. It differs from the host centres by being the centre that executes the global job scheduling policy, directing the events to the servers that will actually execute the job.

D. Task scheduler

The task scheduler is a central component in any simulator of computing systems. Its relevance is even higher in grid simulation, since grids have their performance severely affected by the scheduling policy that is applied. For this reason, it would be very useful to have the capability of evaluating a grid environment under different scheduling policies, without having to create a new model, or to program the policy, for each new configuration.

This capability is offered in iSPD through two distinct options: a library of previously programmed policies, and an interface to generate new policies. The latter option uses an interface to insert the characteristics of a given scheduling policy, including sorting rules and resource allocation policies. A code in an intermediate language is generated from this data. This code is later translated into a Java class, which is compiled and added to the schedulers available in the simulator.

In iSPD the task scheduler is one of the two modules in the simulation engine; the other one being the service centre manager. Figure 3 presents the UML classes diagram

for these modules, where the interface “Master” controls the service centres (queue network) while “CSMaster” controls the scheduler (Workqueue algorithm, in the figure). This implementation allows for a change of the scheduling policy simply by choosing a distinct policy during modelling.

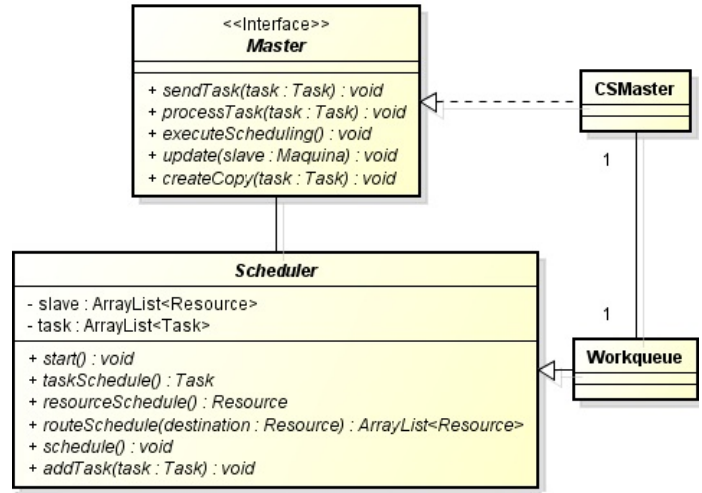


Fig. 3. Scheduling in the simulation engine

This concludes the description of the main components of iSPD, including the component presented in this paper, that is the Task Scheduler. The following section describes the tests performed with different schedulers in order to evaluate how iSPD accommodates this into its simulations, that is, how accurate is the simulations performed by iSPD.

III. TESTS

In order to evaluate the accuracy of the iSPD, several conformation tests were applied. To save space in this paper, we will not address tests related to the language translation processes, only to those concerning the simulation engine. The tests involved a simulation of cluster models and a comparison of those with results measured in a real cluster, and also with simulations executed with Simgrid. This section is organized with a description of the test environment, followed by a discussion about the actual results.

A. Environment

The tests involved a real program running on a research cluster, named cluster-GSPD, and comparing its results with simulations from iSPD and Simgrid. The cluster ran on Debian linux, version 2.6.26, and the tested program was written in C with MPI library (openmpi). The hardware is composed of a front-end plus eight nodes of Pentium dual machines, with 2 Gbytes of RAM. Figure 4 is a schematics of how the cluster is structured. The processing speed of each node of this cluster was measured and the average speed was around 700 billion instructions per second (700,000 MIPS). This value was used by the simulator as the average computational speed for the system.

Besides comparing iSPD and the real cluster, the same problem was modelled and simulated using Simgrid [5], which

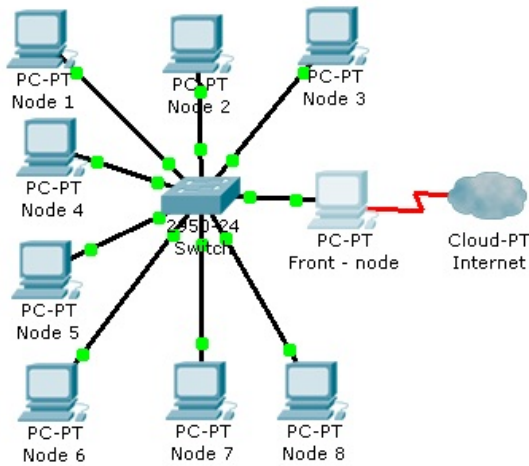


Fig. 4. Cluster-GSPD schematics

was developed by Henri Casanova in 1999, and was initially aimed to study the impact of centralised scheduling policies in distributed and heterogeneous environments. In these tests, the Simgrid's current version (3.6), which is available for Windows, Linux and MacOS systems [10] was used. Although the tests presented here involve only two distinct scheduling policies, Round-Robin and Workqueue, tests with other policies present in iSPD have shown the same pattern of accuracy.

B. Tests with the Round-Robin scheduler

In the round-robin policy, the hosts are organized in a circular list and the tasks are allocated, in their arrival order, to the next available resource. After each allocation, the list is updated and the process continues until all the tasks have been allocated. This policy was implemented in the cluster by a MPI program, whose behaviour is described in figure 5. It is composed by a master process, running in the front-end, with eight slaves, one in each of the cluster nodes. The master process creates tasks, distributing them following the round-robin policy. Each slave has three specific threads: receiving data, munching data and sending back results.

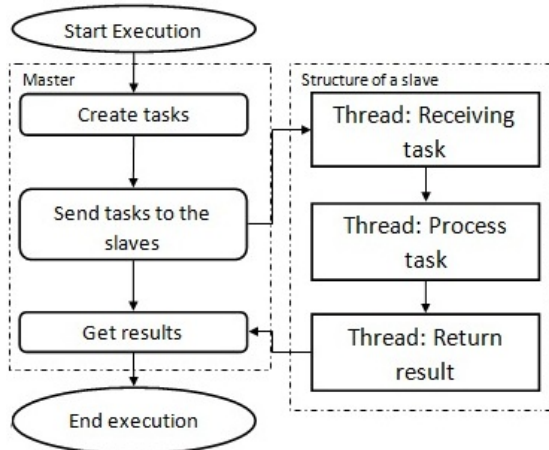


Fig. 5. Testbed program running in the cluster

TABLE I
SIMULATION RESULTS FOR MODELS USING THE ROUND-ROBIN SCHEDULER

system	# of tasks	Time (seconds)	% of cluster
cluster-GSPD	20	1.819	—
iSPD		1.552	85.32
Simgrid		1.511	83.07
cluster-GSPD	40	2.831	—
iSPD		2.592	91.54
Simgrid		2.513	88.75
cluster-GSPD	60	4.306	—
iSPD		4.050	94.06
Simgrid		4.016	93.27
cluster-GSPD	80	5.308	—
iSPD		5.090	95.90
Simgrid		5.018	94.53
cluster-GSPD	100	6.797	—
iSPD		6.550	96.37
Simgrid		6.521	95.94
cluster-GSPD	120	7.807	—
iSPD		7.590	97.23
Simgrid		7.523	96.37

Models for the environment just described were created both in iSPD and Simgrid. For each test there were variations in the number of tasks and their computing and communication costs. The plot in figure 6 presents the results achieved with Simgrid, iSPD and cluster-GSPD when the number of tasks changed. For this test each task has a computing cost of 384,45 Mflops and a communication cost of 1 kbits. Tests were performed with the number of tasks ranging from 20 to 120, in 20 task steps.

As can be seen, the results achieved with iSPD are very close to those provided by Simgrid. In fact, the difference between them was 1.6% on average. The margin of error, when compared to the actual measurements from the cluster-GSPD, was a little higher, with an average error of 6.6%. This is indeed an excellent result since Simgrid had an error of 8.0%. Another aspect that confirms the accuracy of iSPD is that the error linearly decreased when the number of tasks increased, reaching 2.8% for 120 tasks. These results, also summarized in Table I, are a strong indicator that models generated by iSPD are quite accurate and can easily map real environments.

The results presented by the plot in figure 7 are from tests changing the computing cost for the tasks. For these tests, a set of 50 tasks was created, with communication costs varying from 1 kbits up to 40 kbits. The task sets were classified accordingly to their computing costs:

- Small: 384,45 - 1922,25 Mflops
- Medium: 1922,25 - 11533,5 Mflops
- Large: 11533,5 - 38445 Mflops

As in the previous test, the results achieved by iSPD are well correlated to the times measured in the cluster. The overall behaviour follows the same accuracy pattern identified in the previous test, that is, accuracy increases for larger set sizes. In this case, however, the set size is represented by the size of the tasks being simulated, not the number of tasks, which was fixed at 50. Even with this small number of tasks, the error remained under 8% for the smaller tasks, which is very acceptable for simulations. This is, indeed, an interesting

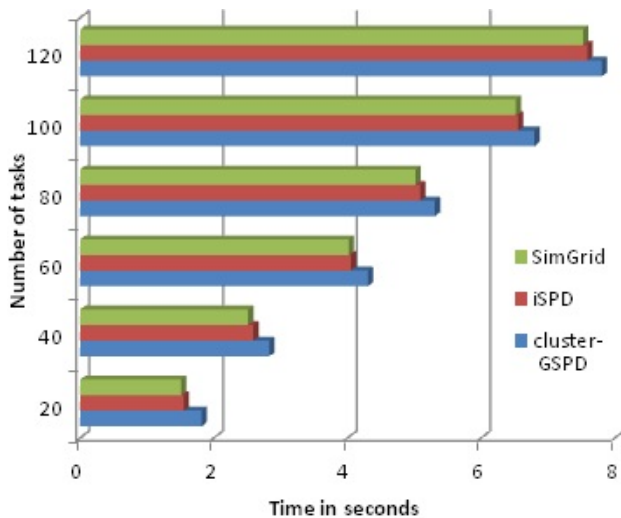


Fig. 6. Measured and simulated times for different number of executed tasks (Round-Robin) [7]

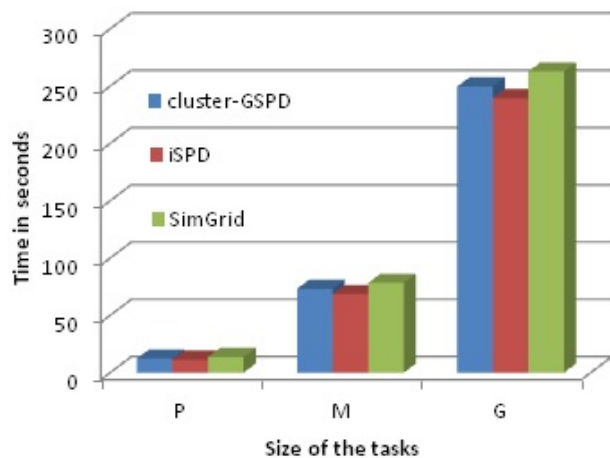


Fig. 7. Measured and simulated times for different computing costs (Round-Robin)

result, showing that iSPD's accuracy does not strongly depend on the type of the tasks being evaluated, but instead depends, mostly, on the amount of processing that is simulated. A remark must be made about Simgrid's results, which presented a different behaviour from the previous test. This difference can be linked to the heterogeneity in the communication costs, which cause extra delays in the simulated execution.

C. Tests with the Workqueue scheduler

The Workqueue scheduler is the second algorithm already implemented in iSPD. In the Workqueue algorithm, tasks are submitted to individual hosts in a bag-of-tasks approach, that is, a host receives tasks as soon as it becomes available. It differs from Round-Robin by the moment that a task is allocated to a host, since here, the allocation occurs only when the host is available, while in Round-Robin the allocation to the next host occurs when a task arrives in the system. The same tests applied to Round-Robin were applied to the Workqueue scheduler.

The results that were achieved confirm the analysis just presented. Figure 8 shows the results when the number of tasks range from 20 to 120. In this plot, it is possible to see that both simulators are reasonably accurate and that iSPD performed better than Simgrid (average error of 3.0% against an average error of 8.2% for Simgrid). As expected, both simulators have higher accuracy when more tasks are simulated.

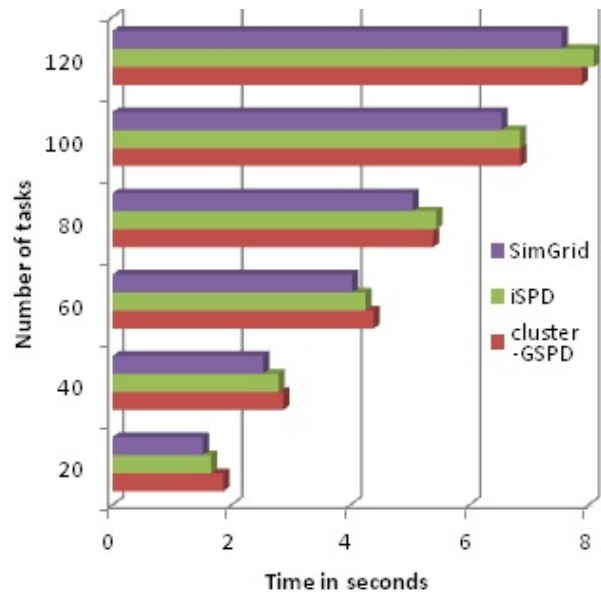


Fig. 8. Measured and simulated times for different number of executed tasks (Workqueue)

Simulation with the Workqueue algorithm was also evaluated with tasks of different computing costs. Achieved results, shown in Figure 9, were also similar to the Round-Robin algorithm, with iSPD being slightly more accurate than Simgrid. For smaller tasks, the error is also larger.

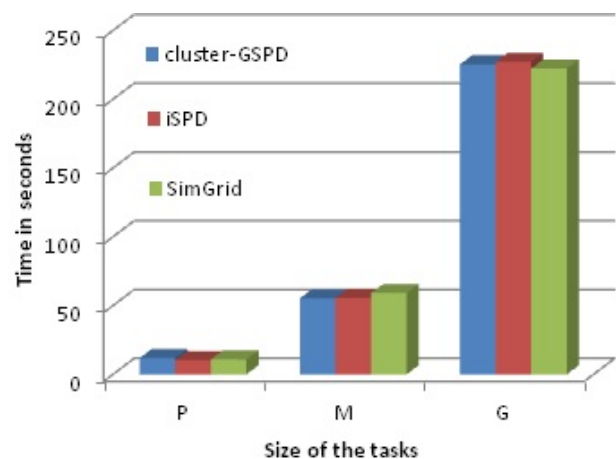


Fig. 9. Measured and simulated times for different computing costs (Workqueue)

IV. RELATED WORK

Several grid simulators have been proposed during the past decade. The most representative are Simgrid [5], and

Gridsim [6]. We now briefly describe some of these proposals, comparing them with iSPD.

a) *Simgrid*: is the first proposed simulator and is still one of the most used. Its initial goal was the evaluation of centralized scheduling policies for heterogeneous and distributed computational environments. New versions of Simgrid have continually been released, although it still lacks an easy-to-use interface to create models.

b) *GridSim*: is another largely used simulator, currently in its 5.0 version. It allows the modelling of different classes of environments, including schedulers and machines. It is based on the SimJava simulation engine. It is quite flexible and has an interface that makes it easier to model several types of computing grids.

c) *GangSim* [11]: was developed to evaluate scheduling policies in grid environments. It allows the analysis of the interaction between local and global schedulers. This feature is very interesting and is under development in iSPD. Gangsim, as the other simulators, does not enable an easy modelling interface, demanding the writing of scripts in an internal language.

d) *OptorSim* [12]: was initially developed to evaluate dynamic replication algorithms used to optimize data location over the grid. This project has been mostly used in evaluations of data replication techniques, which is a different application field when compared to Simgrid, GridSim or other simulators. The major differences to iSPD are that our simulator currently does not address data location and that OptorSim does not have a simple interface to model grids.

e) *BeoSim* [13], [14]: is a discrete event simulator aimed at computer grids assemblies such as Beowulf clusters, interconnected through a dedicated network. It enables the evaluation of smaller grids under different workloads and scheduling policies. It offers a GUI to do part of the simulation process but is the only simulator that is not open source.

f) *GSSIM (Grid Scheduling Simulator)* [15], [16]: was built over GridSim aiming to solve the problems with workload generation and scheduling levels present in other simulators. Like the other simulators presented here, it suffers from the need to model the grid using script schemes.

These simulators have been mostly used to evaluate scheduling policies. In order to perform such evaluation, it is necessary to model the grid (hosts and networks), the workload, and the scheduling policies themselves. The tool presented here, iSPD, makes all these tasks easier to be performed, when compared to other simulators, while providing comparable accuracy.

V. CONCLUSIONS

Simulation is a very powerful aid for the research of scheduling policies and performance evaluation. It can be even more powerful if its application can be simplified for users that are not well versed to programming, as it is the case with typical computing grid users. This simplification is achieved through iSPD, which offers an intuitive iconic interface to model grid environments. The verified accuracy of iSPD, as presented in section III, shows that this approach can provide interesting results without prejudicing precision.

Since the impact of scheduling policies is rather important to achieve optimal performance in distributed systems such as computer grids, their easy modelling should be one of the goals in their simulators. This task is the focus of this paper and can be easily done through iSPD's modelling interface.

Future improvements in iSPD include the implementation of a built-in workload database, which can be used as testbeds for specific grid environment configurations, and the addition of a more versatile interface for results presentation, including options to choose specific metrics. Current efforts also involve the implementation of interpreters for models written for other grid simulators (Simgrid, already functional, and GridSim).

VI. ACKNOWLEDGEMENTS

We wish to acknowledge the support from FAPESP (grant 2008/09312-7) and the Graduate Provost scholarship from UNESP. We also want to acknowledge the invaluable support of V. Aoki, A. Guerra, M.A. Garcia and P.H. Oliveira, former students that worked on the initial iSPD prototypes, and David R. Mercer for language review.

REFERENCES

- [1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *The International Journal of Supercomputer Applications*, vol. 15, no. 3, pp. 200–222, 2001.
- [2] I. Foster and C. Kesselman, *The Grid 2: blueprint for a new computing infrastructure*. Morgan Kaufmann, 2003.
- [3] A. S. Tanenbaum and M. V. Steen, *Sistemas Distribuídos Princípios e Paradigmas*, 2nd ed. Ed. Pearson, 2007.
- [4] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, 2nd ed. John Wiley & Sons, 1991.
- [5] H. Casanova, "Simgrid: a toolkit for the simulation of application scheduling," in *Proc. of the First IEEE/ACM Intl Symposium on Cluster Computing and the Grid (CCGrid 2001)*, 2001, pp. 430–437.
- [6] R. Buyya and M. Murshed, "Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Pract. and Exper.*, vol. 14, pp. 1175–1220, 2002.
- [7] A. Manacero, R. Lobato, P. Oliveira, M. Garcia, A. Guerra, V. Aouqi, D. Menezes, and D. da Silva, "iSPD: an iconic-based modeling simulator for distributed grids," in *Proc. of 45th Annual Simulation Symposium, ANSS'12*, vol. CD-ROM. The Society for Modeling & Simulation International, 2012, pp. 1–8.
- [8] GSPD, "Unesp's parallel and distributed systems research group," 2012, web page available at <http://www.dcce.ibilce.unesp.br/spd>.
- [9] J. Banks, J. S. Carson, D. M. Nicol, and B. L. Nelson, *Discrete-Event System Simulation*, 3rd ed. Prentice-Hall, 2001.
- [10] H. Casanova, L. Legrand, and A. Marchal, "Scheduling distributed applications: The simgrid simulation framework," in *Proc. of the 3rd IEEE Intl Symp. on Cluster Computing and the Grid - CCGrid'03*. IEEE Press, 2003.
- [11] C. Dumitrescu and I. Foster, "Gangsim: a simulator for grid scheduling studies," in *Proc. of the 5th IEEE Intl Symp. on Cluster Computing and the Grid - CCGrid'05*. IEEE Press, 2005, pp. 1151–1158.
- [12] W. Bell, D. Cameron, L. Capozza, A. Millar, K. Stockinger, and F. Zini, "Simulation of dynamic grid replication strategies in optosim," in *Proc. of the ACM/IEEE Workshop on Grid Computing*. Springer-Verlag, 2002.
- [13] W. Jones, L. Pang, D. Stanzione, and W. Ligon III, "Characterization of bandwidth-aware metaschedulers for co-allocating jobs across multiple clusters," *J. of Supercomputing*, vol. 34, pp. 135–163, 2005.
- [14] W. M. Jones, J. T. Daly, , and N. DeBardleben, "Impact of sub-optimal checkpoint intervals on application efficiency in computational clusters," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 276–279.
- [15] K. Kurowski, J. Nabrzyski, A. Oleksiak, and J. Weglarz, "Grid scheduling simulations with gssim," in *Proc. of the 13th Intl Conf. on Parallel and Distributed Systems - Volume 02*, 2007, pp. 1–8.
- [16] GSSIM, "Grid scheduling simulator website," 2012, web page available at <http://www.gssim.org>, last accessed in January 2012.