A tool for model conversion between simulators of grid computing

Gabriel C. Furlanetto
Paulista State University - UNESP
Computer Science and Statistics Dept, Rio Preto
gcovfur@gmail.com

Rafael S. Stabile
Paulista State University - UNESP
Computer Science and Statistics Dept, Rio Preto

Aleardo Manacero
Paulista State University - UNESP
Computer Science and Statistics Dept, Rio Preto
aleardo@sjrp.unesp.br

Renata S. Lobato
Paulista State University - UNESP
Computer Science and Statistics Dept, Rio Preto

Denison Menezes
Paulista State University - UNESP
Computer Science and Statistics Dept, Rio Preto

Roberta Spolon
Paulista State University - UNESP
Computing Dept, Bauru

ABSTRACT
High performance computing systems usually have a high usage cost, even with the use of shared systems, such as grids and clouds. To reduce such costs it is necessary to optimize system's utilization through performance analysis of one's application. This can be done through simulators designed specifically for performance prediction/analysis, such as GridSim and Simgrid, among several grid computing simulators developed in the past years. Unfortunately, there is no compatibility between these simulators, making models developed for one of them unusable for the others. This is even more problematic when one notices that the models for these simulators demand coding of several complex functions, making model reuse hard to achieve. In this paper we present an extension made to iSPD (iconic Simulator of Parallel and Distributed Systems), which is already an easy-to-use grid simulator, in order to enable model conversions. With this extension an user is able to read models written for GridSim or Simgrid, converting them into the iconic models used by iSPD. He/she is also able to convert iSPD models into models for those simulators. The process of conversion is discussed, showing the actions necessary to convert the models. A comparison between the simulation of models built using each simulator is presented.

Author Keywords
model reuse; model interoperability; iconic modeling; grid computing simulation

ACM Classification Keywords
I.6.7 SIMULATION AND MODELING: Simulation Support Systems

ANSS 2015 April 12-15, 2015, Alexandria, VA

INTRODUCTION
High performance computing has been widely used to solve problems of fields ranging from pure science to commercial applications. Supercomputers have been used to provide high performance for a long time [1], but their high cost made a way to more affordable solutions, such as sharing computing resources through computer grids. Although more cost-effective, grids have some constraints, such as communication latencies, that make it hard to achieve performance with naive solutions. Besides that, grid computing systems have been largely used by biologists, physicists, chemists, and other users that are not programming-literate.

To improve the performance of an application, a developer/analyst can make use of performance analysis tools. Among these tools one finds grid simulators, such as SimGrid [2], GridSim [3], GangSim [4], and iSPD [5]. The use of simulation is preferable because simulations can be performed offline, avoiding the consumption of grid resources to collect performance data.

A trouble with most simulators is that they demand some programming effort from the person that uses it, while creating a model for a grid/application. In fact, except iSPD (**i**conic **S**imulator of **P**arallel and **D**istributed systems), which is completely iconic, all simulators just listed need some programming, either in C, Java, or script languages, besides partial graphical interfaces. This fact is a problem to most of the grid computing users, who do not have a strong, frequently not even a weak, programming knowledge. The development of iSPD was driven towards these users.

Another issue with the effort needed in the modeling process is that modifying a model is not usually simple.

Modifying, or reusing, models is interesting because reuse typically reduces the amount of time needed to model development, and to alternative evaluation, for the system. Besides that, there are no simple mechanisms that allow reuse of models built for the same simulator. This is even worse if one tries to reuse a model built for a different simulator.

With iSPD, model reuse is simpler, since the models are easily built with its iconic interface. We also added a component that allows conversions from/to the major grid simulators (GridSim and SimGrid, which have more than 1,000 citations each (Google Scholar)). The motivation for this is that there is already a large amount of models built for those simulators, which could be reused by iSPD users. Reusing a model is interesting because one can make simulations starting with a model previously verified, being able to easily adapt it through iSPD's iconic interface.

In this paper we present how iSPD enables model reuse and the results achieved with these conversions. In the following sections, we first present an overview of SimGrid, GridSim, and iSPD, before presenting the architecture for model conversions implemented in iSPD. Validation tests for these conversions, and therefore, model reuse, are also presented.

SIMULATORS OF GRID COMPUTING

One can find some simulators of grid computing in the literature. Among them, the most referenced ones are GridSim and SimGrid, both developed around year 2000. As already indicated, models built for one of them cannot be used in the other, which can be seen as a waste of resources. In the following paragraphs, one finds the description of architectural aspects of GridSim and SimGrid, as well as iSPD.

GridSim

GridSim was introduced by Buyya and Murshed [3] and its current version is the 5.2 [6]. It was developed in Java and has some interfaces enabling to partially build models through graphical interfaces. Some of the modeling has to be made by Java programming. Therefore, a GridSim model is a set of packages in Java, describing the grid, the tasks and the meta-schedulers.

Among the features provided by this simulator, one finds functions to manage node failures, time reservations, and parallel tasks. It also permits modeling heterogeneous tasks, either CPU-bound or I/O-bound [7]. Other extensions of GridSim include the support for data grids [8], and a new tool, called CloudSim, to model cloud computing [9].

It has a layered architecture. The first one is a basic Java interface. Above it, one finds the discrete event manager, which carries out the simulation. The third layer contains tools to help the model creation. The fourth layer contains models for the meta-schedulers that will be used in the grid. Finally, the fifth level is concerned with users, data input/output and the modeled applications.

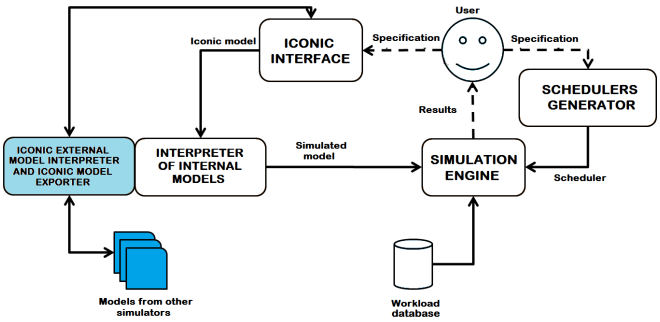

Figure 1. Main modules in iSPD's architecture.

SimGrid

SimGrid was introduced by Casanova [2], being currently in the version 3.11.1 [10]. It was developed in C and has an interface that enables the modeling of specific applications, such as P2P or MPI. To model grid environments, the user has to write code in C, or Java for part of the model.

Its features include the capability to model background traffic and computation. It is also very interesting that it can directly simulate MPI programs, which may be seen as a consequence of having models built directly in C language.

As GridSim, this simulator has also a layered architecture. In SimGrid's case, the layers appears as extensions from previous versions, where new functionality is added. In its current version it has three major layers: a modeling layer, containing several components, a simulation interface (SURF) and a model library for SURF.

iSPD

iSPD was introduced by Manacero et al [5], being currently in the version 2.0 [11]. It was developed in Java and has an interface that allows for easy modeling of grids, including the meta-schedulers. Its iconic interface enables modeling hosts, clusters, communication links, users, schedulers and a variety of BoT (Bag of Tasks) workloads. Comparing to the other simulators, it offers a better modeling interface, an equivalent accuracy, and it is as fast as the faster ones.

It has a modular architecture, shown in Figure 1. Its main modules include the modeling interface, the simulation engine and the model converter. These and the remaining modules are described next.

- **Iconic interface:** it provides a graphical interface for grid modeling, providing an iconic process to create models, including another module to visually create meta-schedulers, if necessary.

- **Simulation engine:** it is the module in charge of creating a queue network, including service centers. It also provides all the visualizations.
- **Model interpreter:** this module, which is the extension presented in this paper and it is shaded in Figure 1, does the conversion of models built for GridSim or SimGrid to iSPD and of models built for iSPD to GridSim or SimGrid models.
- **Trace database:** it comprises a set of functions to manage workload traces, including the creation of traces from simulated environments.
- **Scheduler generator:** it provides an interface to manage and create meta-schedulers and local schedulers. Its graphical interface offers some native schedulers as well as GUIs that allow users to formulate their own scheduling policies [12].

Particularly, including a module that performs model conversions allows model reuse. Reusing models built for a different simulator is interesting because one can retrieve a model built to GridSim, e.g., and easily modify and simulate it with iSPD. The other way around is also possible, although it is actually uninteresting due to iSPD's advantages.

As the other grid simulators, the iSPD's project is continuously evolving. Currently there are extensions being developed to allow simulations of data grids, cloud computing and dependent tasks, modeled by directed acyclic graphs (DAG).

MODEL CONVERSION IN ISPD

As explained before, the module that performs model conversions was introduced to allow the reuse of previously written models by iSPD. This can be considered as an advantage because any user that has a model for an older grid system can quickly model a new one using the former as a starting model. Reusing a model within GridSim and SimGrid's architectures involve code rewriting and insertion, which is difficult for non-expert users. Doing this through iSPD involves reading the older model and creating an iconic model in iSPD's modeling interface. Then, this iconic model can be easily modified, improved or grown, through this interface, allowing for the simulation of the new model quite shortly.

The process of model conversion is equivalent to code compiling/interpreting. The source code for a given simulator is the input of the conversion module, which produces an "executable" code in iSPD's iconic language. Therefore, the development of this module was based in the specification of a set of grammatical rules to read/write specific languages. These conversions are described in the following paragraphs.

Importing a model

To convert a model written for GridSim or SimGrid, it was necessary to develop two conversion grammars, one for each simulator. These grammars were used to enable the parsing procedure shown in Figure 2. In that figure one can see that the parser creates a list of elements, which are iSPD's icons, and then maps these elements to specific structures by a conversion table, partially shown in Table 1.

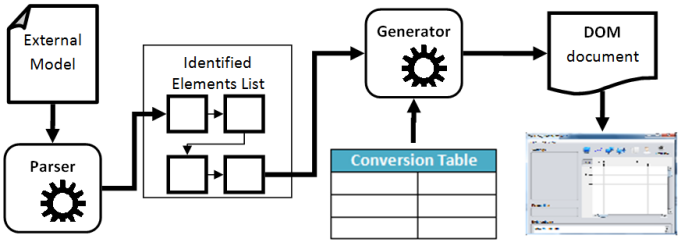

Figure 2. Importing an external model.

In GridSim's grammar, which has two rules described below in the Backus-Naur form (BNF), the focus was in translating Java to iSPD's modeling language, called **iconic modeling standard** (*imsx*). The rules for GridSim are defined, for example, in the form:

< CLASS > ::= "class" , or

<import_declaration> ::= <IMPORT> [<STATIC>] <name> [<DOT>"*"] <SEMICOLON>

Where, in the first one, the parser identifies that a Java class will be defined. In the second rule it is defined how an **import** of a class has to be declared and which tokens had to be expected.

The set of rules defined for GridSim conversion is different from a typical compiler's front-end parser since it will not generate Java bytecodes. Also, the parsing should not be performed over every line of code in the file storing the model since only specific lines are actually related to the model. In the process depicted in Figure 2 the foreign model is read by the parser, who creates a list of specific tokens (as the ones shown in Table 1). These tokens are linked with templates of icons in iSPD, having their values associated with specific parameters in the icon.

For SimGrid the parsing process is also simple. In this case it is oriented by the XML tags used in SimGrid's models, which come from two files, one containing data about hosts and links and other with the description of their roles (master/slave, for example). It must be noted, however, that the conversion process currently does not make the parsing of the meta-schedulers coded in C files for SimGrid. To overcome this, the user converting SimGrid models has to use iSPD's scheduler GUI to generate equivalent meta-schedulers, which can be done easily. To model the schedulers the analyst may either use a native scheduler or create a new one. The creation process is guided by few windows where the user can specify the rules that the scheduler will follow.

Table 1. Mapping of specific parameters of foreign simulators to iSPD models

SimGrid	GridSim	iSPD
”CPU”	machine	host icon
”network_link”	SimpleLink	communication icon
–	Route	communication icon
function=”slave”	machine	host icon as slave host
function=”master”	GridResource	host icon as master host
function=”master” first argument	gridlets	tasks (workload)

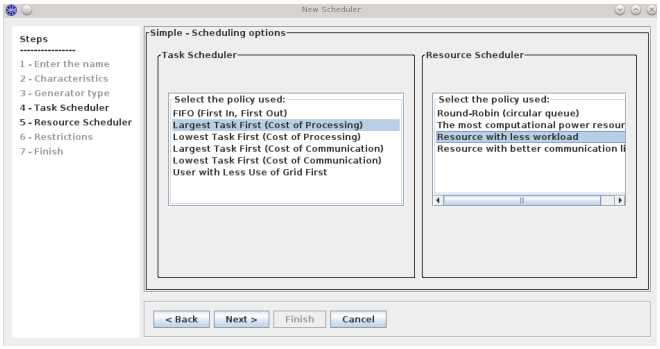

Figure 3. Modeling a trivial scheduler with iSPD's interface

Figures 3 and 4 show two of these windows, one creating a simpler, guided model, and the other creating a scheduler through a mathematical formulation.

For both sources the process of model conversion is coordinated by mapping the identified elements to iSPD icons. This procedure involves reading data arguments in the source's model and mapping them to the data arguments appearing in each icon.

Exporting a model

Besides it is not the primary goal for the conversion module, it is also possible to export models created with iSPD to models runnable by GridSim or SimGrid. It should be noted that the developers believe that this conversion is actually unnecessary since iSPD provides a better modeling interface, accurate simulation results and simulation speeds comparable to any other simulator. It has been included in the model converter just to make the system complete[1].

The exporting process was also developed by the definition of two conversion grammars, one for each foreign simulator. The steps towards the conversion are shown in Figure 5, where an important part is the conversion table, partially shown in Table 2.

As it would be expected, the whole conversion starts with the matching of the relevant icons in iSPD's model

[1] And, of course, to allow a fanatic user of GridSim/SimGrid to compare the models

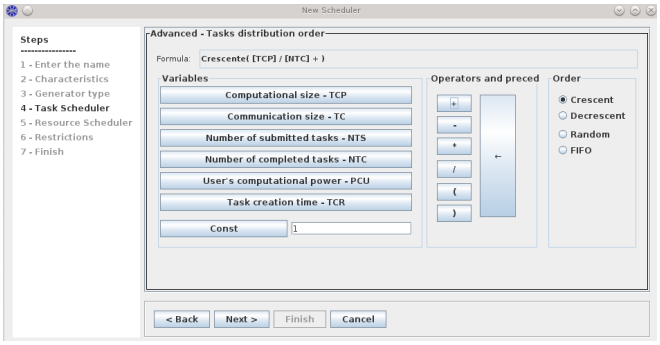

Figure 4. Modeling a complex scheduler with iSPD's interface

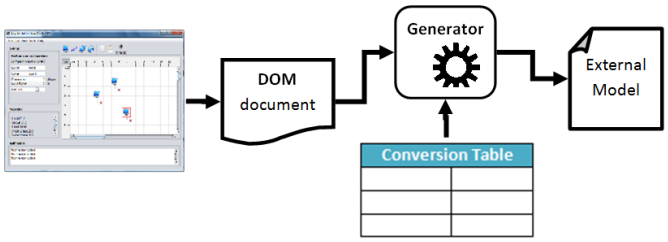

Figure 5. Exporting an iSPD model

and their association with templates for a given simulator. If one is converting to SimGrid the process will create two files, one with hosts/links description and other with their relationships and actions. For conversions to GridSim a single file is created, containing the whole model.

It is important to notice in Table 2 that the code for SimGrid's meta-scheduler is not generated. This was a design option, since in SimGrid the schedulers are hand-coded in a separated file, which is used only at simulation time. To create this file it is necessary to have a Java to C converter, demanding additional features to iSPD that were not a major concern. Therefore, for SimGrid users it is necessary to manually write their schedulers.

Modeling comparison

After implementing the model conversion components in iSPD we can compare what each simulator can model and which model parts can be converted. Table 3 shows some characteristics of grid systems that can be modeled by SimGrid, GridSim and iSPD. From this table it is possible to verify that none of the simulators are able to model all characteristics. As the conversion process conversion is concerned we see that iSPD can convert the most relevant characteristics.

The characteristics that are not directly convertible include some that can be modeled only by iSPD (*Internet*, *Cluster* and *Scheduler Manager*). They also include two (*Holiday Load* and *Operating System*) that appears only in GridSim. These were not included because to distinguish loads by the day that they occur can be made through trace files, instead of controlling the calendar,

Table 2. Mapping of iSPD icons to parameters in foreign simulators

iSPD	SimGrid	GridSim
host/cluster icons	”CPU”	machine
internet icon	”network_link”	Router
communication icon	”network_link”	SimpleLink
tasks (workload)	mapped to specific functions	Process
schedulers	manually coded	copied from iSPD library

Table 3. Model components in the analyzed simulators. The last column indicates which components can be converted to iSPD models.

Component	GridSim	SimGrid	iSPD	Convert
Single Host	yes	yes	yes	yes
Link	yes	yes	yes	yes
Internet	yes	no	yes	no
Tasks	yes	yes	yes	yes
Cluster	no	no	yes	no
Grid	yes	yes	yes	yes
User	yes	no	yes	yes
Holiday Load	yes	no	no	no
Operating System	yes	no	no	no
Scheduler Manager	no	no	yes	no

and that we believe that identifying specific operating systems is more relevant for cloud systems. Therefore, no further modifications were necessary in the remaining components of iSPD.

CONVERSION EVALUATION

To evaluate the conversion process for each foreign simulator it is necessary to compare results from seven different versions of the same model. The versions are the native model built for iSPD and for the foreign simulators, and the models obtained from the import and export operations. These models are named “iSPD”, “GridSim”, and “SimGrid” for the respective native models; “ImportedSimGrid”, and “ImportedGridSim” for the imported models; and “SimGridExported”, and “GridSimExported” for the exported ones. The process of importing/exporting models is started through the iSPD’s menu, as shown in Figure 6, which presents the tabs to import a SimGrid’s model.

Although several different grids were modeled, and tested with similar results, only three grid models are presented here. These grids were selected because they represent different conditions for the simulators. A first grid, name Grid A, is a simple homogeneous system, with a single meta-scheduler. The second grid, Grid B, expands it through a two-level meta-scheduler,

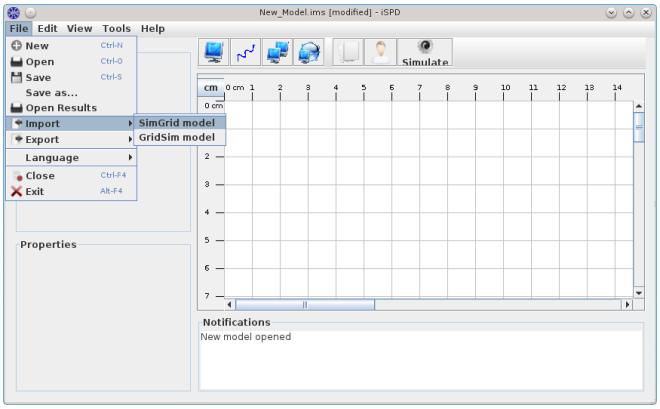

Figure 6. Starting an import operation in iSPD

keeping the system homogeneous. The third one, Grid C, introduces a larger number of working hosts as well heterogeneity among them.

For each model we conducted tests with a variable number of tasks, ranging from 2,000 to 16,000 tasks, which represent a reasonable amount of grid occupation. In all three grids the meta-schedulers used a round-robin algorithm to allocate tasks to hosts, as it is found in many systems. The results presented here are the average of 15 runs by each simulator, giving the necessary statistical stability from a mean convergence test.

Tests for Grid A

This grid has only three working nodes and a fourth node running the meta-scheduler, as shown in Figure 7. In this representation it must be noted that the network connections were introduced to follow GridSim’s restrictions for interconnections. The working nodes have a processing speed of 50,000 MFlops and all the communication links have a bandwidth of 1,000 Mb/s.

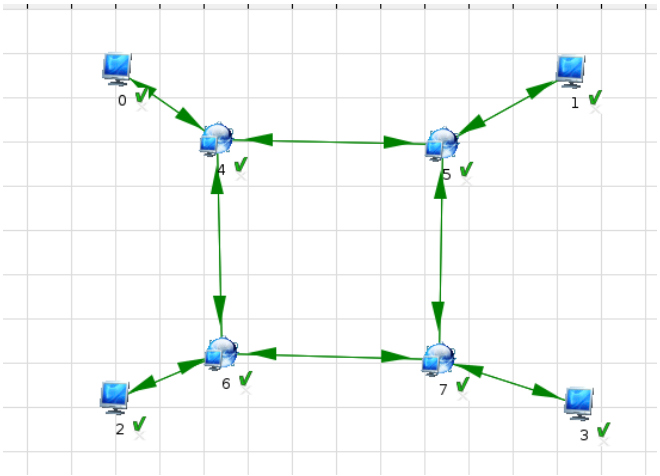

Figure 7. Model for Grid A

The simulations made with this model resulted in very similar results for all models and simulators. The plot presented in Figure 8 shows these results, where one

Table 4. Simulated execution times for grid A for each model (seconds)

Simulated Model	Number of tasks 4000	16000
iSPD	16711	66730
SimGrid	17407	67575
GridSim	15825	65812
ImportedSimGrid	16710	66778
SimGridExported	17407	67575
ImportedGridSim	16714	66721
GridSimExported	16020	66237

can see that the simulated execution times produced by models ran on SimGrid were higher than the average for the other simulators (6% higher for 4,000 tasks and 2% higher for 16,000 tasks). These results can be better evaluated in table 4, where the times are measured in seconds.

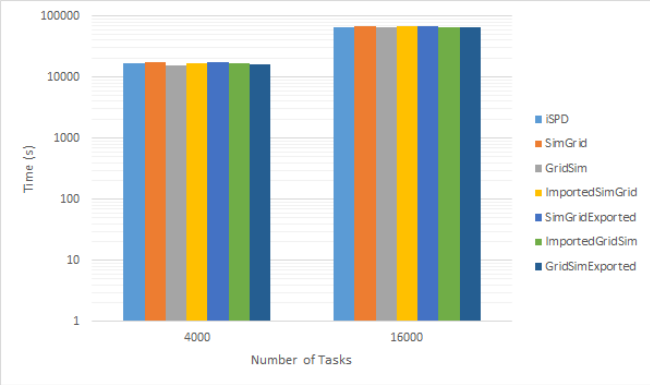

Figure 8. Experimental results from the simulation of all converted models

Tests for Grid B

The Grid B uses a two-level procedure to schedule tasks among hosts. A top-level meta-scheduler allocates tasks to two second-level (intermediate) masters. These intermediate hosts allocate tasks to a set of five homogeneous slave hosts each. This grid is shown in Figure 9. The working nodes have a processing speed of 50,000 MFlops and all the communication links have a bandwidth of 1,000 Mb/s.

The simulated results from all models are shown in Table 5, being better visualized through Figure 10. From these results it is also possible to see that the results for GridSim and iSPD are very similar, both a little below SimGrid. The difference between SimGrid and the other simulators remained around 10%, but it is important to see that iSPD's results, which is our concern, provided the same results as GridSim.

Tests for Grid C

The final test presented here is an expansion in the previous grid, where three additional intermediate meta-schedulers were added, each of them distributing work to five hosts, but with different processing speeds, as

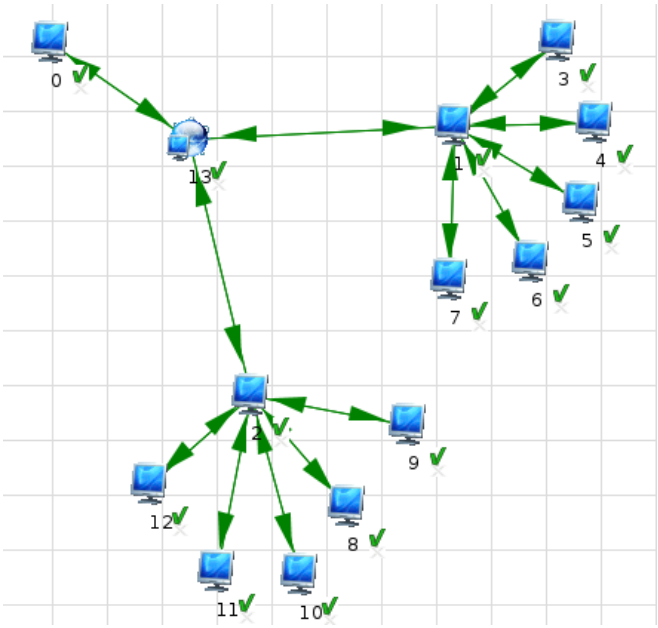

Figure 9. Model for Grid B

Table 5. Simulated execution times for grid B for each model (seconds)

Simulated Model	Number of tasks 2000	4000	8000	16000
iSPD	2535	5047	10062	20086
SimGrid	2819	5613	11221	22416
GridSim	2584	5189	10472	21959
ImportedSimGrid	2538	5043	10063	20085
SimGridExported	2815	5613	11221	22416
ImportedGridSim	2540	5045	10063	20099
GridSimExported	2589	5194	10537	22089

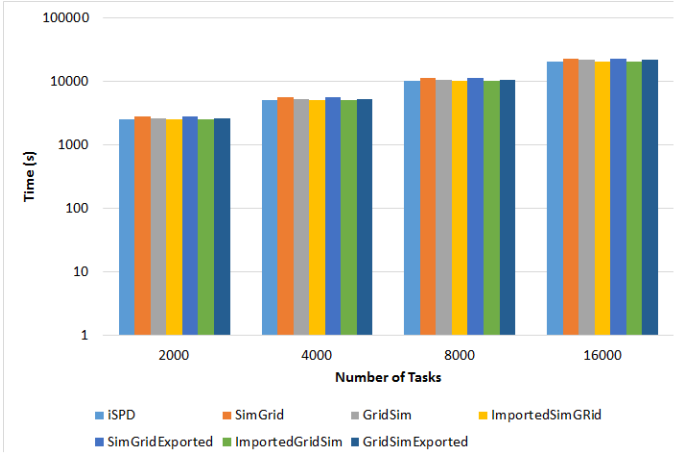

Figure 10. Simulated execution times for Grid B

shown in Figure 11. In this grid two clusters have hosts with processing speeds of 50,000 MFlops, while the remaining three clusters have machines of 50,000, 30,000 and 24,000 MFlops. All links are capable to transfer 1,000 Mb/s.

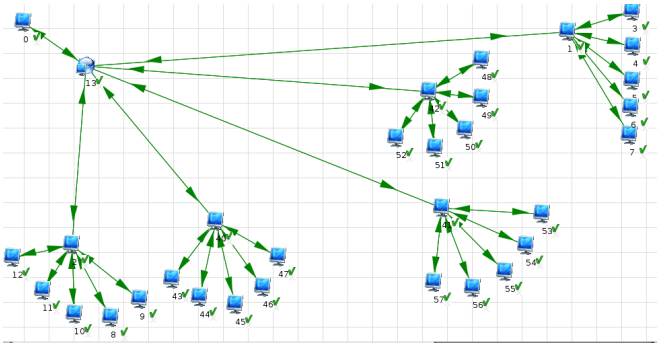

Figure 11. Model for Grid C

Table 6 displays the simulated execution times for each model, which can also be visualized through Figure 12. Once again the results from iSPD and GridSim are closer to each other. SimGrid still produces higher timespans, with the worse results for the smallest set of tasks ($\approx$ 19%). For larger amounts of simulated tasks the difference remained under 10%.

Table 6. Simulated execution times for grid C for each model (seconds)

Simulated Model	Number of tasks			
	2000	4000	8000	16000
iSPD	2112	4222	8405	16767
SimGrid	2564	4680	9398	17791
GridSim	2135	4295	8626	16402
ImportedSimGrid	2114	4238	8412	16776
SimGridExported	2564	4680	9398	17748
ImportedGridSim	2120	4235	8414	16801
GridSimExported	2228	4226	8745	16649

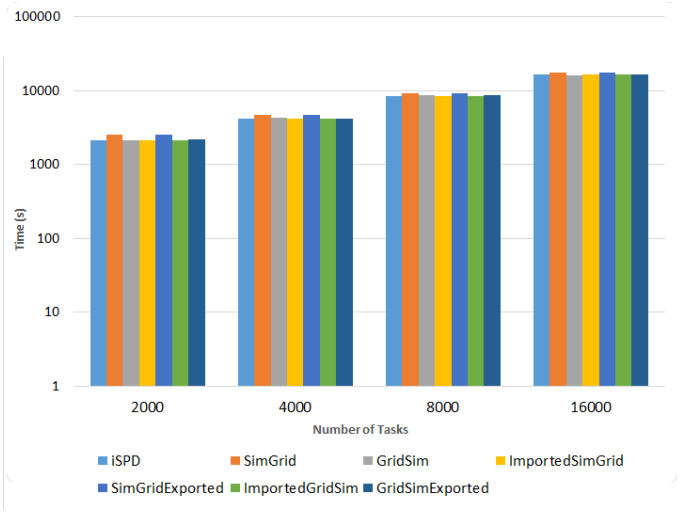

Figure 12. Simulated execution times for Grid C

Performance of the simulators

Another value measured was the time spent by each simulator to produce the results. Results for Grid A were not measured because the simulators, except GridSim, could present results almost instantaneously, even for 16,000 tasks. Partial results (4,000 and 16,000 tasks) for the other two grids appear in Table 7. In all cases it is possible to see that SimGrid and iSPD are much faster than GridSim, and that the increase in the number of tasks does not produce an exponential growth in the time needed to simulate each model, as occurs with GridSim.

Table 7. Time spent to simulate each grid model, separated by simulator (seconds)

Simulated Model	Number of tasks			
	Grid B		Grid C	
	4,000	16,000	4,000	16,000
iSPD	0.011	0.050	0.090	0.115
ImportedSimGrid	0.013	0.049	0.023	0.119
ImportedGridSim	0.012	0.057	0.025	0.098
SimGrid	0.046	0.172	0.067	0.324
SimGridExported	0.046	0.172	0.167	0.334
GridSim	9.933	855.733	12.234	902.340
GridSimExported	10.400	858.667	12.113	934.574

From this table it is possible to verify that iSPD usually produced results faster than SimGrid. In few cases SimGrid was negligibly, just few milliseconds, faster than iSPD. In fact, although the difference remained under a second, it is possible to perceive that it grows for a larger number of tasks, with iSPD becoming three times faster than SimGrid for the larger simulations. Another remark is that iSPD simulate models converted from SimGrid faster than SimGrid itself (lines “IportedSimGrid” and “SimGrid”). These results allow to conclude that iSPD is a good choice for grid simulations and that converting old models built for other simulators is an interesting feature in it.

CONCLUSIONS

The results just presented show that the model conversions between these grid simulators is possible. The simulations performed over the converted models reached equivalent results, independent of being directly built or obtained by conversions. This implies that models formerly built for SimGrid or GridSim can be easily reused by iSPD, while other simulators do not have such feature.

Some important remarks must be made from these results and the operation of these simulators:

1. All models produced equivalent results
2. Models could be built with iSPD more easily and faster than with the other two simulators
3. iSPD produced results faster than the other two
4. Models imported by iSPD produced the same results produced by their original simulators

These observations enable to conclude that the possibility of model reuse, introduced by this new component, is very interesting. It is also interesting to note that an

user can easily modify an older model built for SimGrid or GridSim and simulate new alternatives with iSPD.

A final remark is related to the conversions of SimGrid models. As stated before, these conversions demand a manual codification for the adopted scheduler. Fortunately, this is an actual problem only when exporting models from iSPD, since the schedulers that it generates automatically have to be implemented in C. For imported models, this can be easily solved by iSPD's component for scheduler generation, through its graphical interface to create Java code for the scheduling algorithm.

Next steps in this work would involve the inclusion of conversion procedures for cloud models, specially those from CloudSim, and a converter for the schedulers defined in SimGrid. While the latter is a problem only to export models, the former is a needed feature considering that iSPD's future version will include cloud simulation (PaaS and IaaS) as a new feature.

ACKNOWLEDGMENTS

Authors want to acknowledge the support from FAPESP by the grant 2012/15127-3 (funding the iSPD project) and to CNPq (scholarships provided to Mr. Furlanetto and Mr. Stabile).

REFERENCES

1. Franck Cappello, Henri Casanova, and Yves Robert. Preventive migration vs. preventive checkpointing for extreme scale supercomputers. *Parallel Processing Letters*, 21(02):111–132, 2011.

2. Henri Casanova. Simgrid: a toolkit for the simulation of application scheduling. In *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001*, pages 430–437, 2001.

3. 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.*, 14(13-15):1175–1220, 2002.

4. C. L. Dumitrescu and I. Foster. Gangsim: a simulator for grid scheduling studies. In *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 2 - Volume 02*, CCGRID '05, pages 1151–1158, Washington, DC, USA, 2005. IEEE Computer Society.

5. A. Manacero, R.S. Lobato, P.H.M.A. Oliveira, M.A.B.A. Garcia, A.I. Guerra, V. Aoqui, D. Menezes, and D.T. Da Silva. ispd: an iconic-based modeling simulator for distributed grids. In *Proc. of the 45th Annual Simulation Symposium*, ANSS '12, pages 5:1–5:8, San Diego, CA, USA, 2012. SCS.

6. GridSim. Gridsim's project website. Available at <http://www.cloudbus.org/gridsim/>, October 2014.

7. Kalim Qureshi, Attiqa Rehman, and Paul Manuel. Enhanced gridsim architecture with load balancing. *The Journal of Supercomputing*, 57(3):265–275, 2011.

8. Anthony Sulistio, Gokul Poduval, Rajkumar Buyya, and Chen-Khong Tham. On incorporating differentiated levels of network service into gridsim. *Future Generation Computer Systems*, 23(4):606–615, 2007.

9. Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

10. SimGrid. Simgrid's project website. Available at <http://SimGrid.gforge.inria.fr/>, October 2014.

11. GSPD. Gspd's homepage. Available at <http://www.dcce.ibilce.unesp.br/spd/>, October 2014.

12. D. Menezes, A. Manacero, R.S. Lobato, D.T. da Silva, and R. Spolon. Scheduler simulation using ispd, an iconic-based computer grid simulator. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000637 –000642, july 2012.