Modeling and Simulation of Cloud Computing with iSPD

Diogo T. da Silva[1], **João A. M. Rodrigues**[1], **Aleardo Manacero**[1],
Renata S. Lobato[1], **Roberta Spolon**[2], **Marcos A. Cavenaghi**[3]

[1]Dept of Computer Science and Statistics
São Paulo State University - UNESP, São José do Rio Preto, Brazil

[2]Dept of Computing
São Paulo State University - UNESP, Bauru, Brazil

[3]Humber Institute of Technology and Advanced Learning, Toronto, Canada

`tavareko@gmail.com, {aleardo.manacero, renata.spolon}@unesp.br`

Abstract. *Cloud Computing, enabled by technological enhancements and the trend for reduction of investments in IT's physical infrastructure, is the major computing infrastructure nowadays. However, its heterogeneity makes difficult to know if the use of a given environment is efficient or not. In this context, the performance evaluation of cloud systems is useful both to clients, who need to find the best resource configuration for their applications, and to providers, who need to evaluate which scheduling and allocation policies of resources and virtual machines are most cost effective. Simulation is a good approach for this evaluation since it can be done offline. The known cloud computing simulators have issues related to their usability and modeling capability. This work extends the iconic approach for modeling and simulation offered by iSPD to cloud computing, adding icons for virtual machines and VMMs. Our results show that iSPD is faster than Cloudsim with equivalent accuracy, while providing an easier interface to model and simulate IaaS and PaaS environments.*

1. Introduction

Cloud computing has been widely used since its advent. This use can be credited to the concept of resource provisioning by demand, that is, resources are allocated only when needed, with users paying only for what they actually have used. Although this is a classic approach, its broad usage only became possible by the remarkable improvement of technologies such as multicore processors, hardware virtualization, communication, autonomous systems and grid computing [Buyya et al. 2011]. These technologies were merged to enable cloud computing and to reduce IT maintenance costs. They allowed businesses to avoid building their own datacenters, which are typically designed to accommodate theoretical, but not so frequent, peaks of usage, remaining partially idle most of the time.

Mell and Grance [Mell and Grance 2011] define cloud computing as a pay-per-use model that enables access to the provided resources by demand and through the internet. This access occurs over a set of configurable resources, shared among all cloud clients, and made available with minimal management effort. Although the services offered by cloud systems can be classified into three different classes – IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service) – the major players in the field are mostly concerned with IaaS, with the most known cloud providers, including Google Compute Engine, Amazon Web Services and Microsoft Azure, working with class of service.

Besides the inherent advantages of cloud computing, its use does not avoid actual costs, which need to be optimized. From the client's perspective it is useful to determine the right resource configuration in order to speedup applications while reducing the overall cost. On the other hand, to resource providers, it is interesting to evaluate the policies adopted for resource provisioning, allocation and scheduling in order to minimize production costs (maximizing profit). To better understand the system's behavior, and how the desired goals can be achieved, one may use simulation. Using simulation is preferable to benchmarks because the latter implies possible down-times in order to measure the system running under different conditions or performing non-productive processing.

There are some simulators of cloud computing available, such as CloudSim [Buyya et al. 2009], or iCanCloud [Castane et al. 2012]. These are extensions of either grid or network simulators, keeping their approaches for modeling, which are based on heavy programming of systems characteristics. In this paper we introduce an extension of iSPD (**i**conic **S**imulator of **P**arallel and **D**istributed systems) [Manacero et al. 2012], which is a grid simulator based in iconic modeling, allowing an easy process to create and simulate grid, and now cloud, models.

In the remaining sections we first review some cloud simulators. This is followed by a review of common cloud services, pointing out how they can be simulated. After that, the following sections describe the development of this work and results achieved with cloud simulation using iSPD, and the conclusions drawn from these tests.

2. Related Work

Cloud systems differ from conventional distributed systems due to its definition as a business model. Besides that, most of its infrastructure is similar to regular distributed systems. In this scenario is natural that simulators designed for cloud simulation derive from simulators of distributed systems as occured with CloudSim, iCanCloud, GreenCloud, and others, as briefly described here. Another important aspect is that they are mostly concerned with IaaS since this is where major players work.

iCanCloud [Castane et al. 2012] is a cloud simulator developed to evaluate the cost-performance ratio in IaaS services. It was developed using OMNet++ and INET frameworks, demanding knowledge on these tools to use iCanCloud. It also offers a graphical interface for part of the modeling work. Among its features we can point capabilities to model MPI applications, storage systems and an actual application directly into the simulator.

In another direction GreenCloud [Kliazovich et al. 2010] was developed using the NS2 simulator, aiming to evaluate the energy performance of clouds. It is mostly concerned with communication costs due to the use of NS2. Modeling with GreenCloud is done through scripts and functions written in Tcl and C++.

Buyya et al [Buyya et al. 2011] developed CloudSim from the simulation engine used in GridSim [GridSim 2022]. It allows modeling of datacenters, virtual machines, hosts, and brokers. Modeling is performed with Java coding, where the user can model datacenters, individual hosts, VM provisioning, and scheduling policies. CloudSim served as a framework for several extensions, such as WorkFlowSim [Chen and Deelman 2012], which simulates workflow processes, and iFogSim [Gupta et al. 2017], to simulate fog and edge computing.

Other grid simulators, such as Simgrid [Casanova 2001], have extensions aimed to simulate certain aspects of cloud computing, but are not strictly concerned with virtualization, for example. Still some network simulators, such as NS-3, can be used to program cloud environments, although demanding codification of large parts of the model and being inefficient to model computing loads/tasks.

3. Cloud Simulation with iSPD

The simulators just described have been used regularly in the reported research, specially CloudSim. Their major characteristics/differences are presented in Table 1, which reinforces the fact that modeling poses some constraints to those who are not comfortable with programming. A different approach is offered by iSPD, which was designed having easiness-of-use as a goal. This tool was originally designed aiming the simulation of grid computing and is based in graphical interfaces to build models using representative icons.

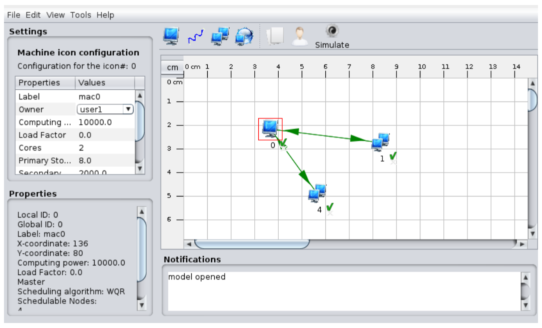
Table 1. Main characteristics of the major cloud computing simulators

	Framework	Programming Language	Graphical interface
CloudSim	GridSim (partially)	Java	Limited (extensions)
iCanCloud	OMNeT++, MPI	C++	Present
GreenCloud	NS2	C++, Tlc	Limited (extensions)
iSPD	none	Java	Present

3.1. How iSPD works

Before presenting iSPD's extension to cloud simulation, it is useful to describe its original version. As pointed out, modeling is based upon graphical interfaces. Basically, users have to model a grid using icons to represent and configure objects, such as computing nodes or network links. The simulation process is performed through the following steps:

1. **Modeling -** user creates the system's topological model, configuring each object with data such as computing power or network bandwidth using the interfaces shown in Figure 1. The data shown in the "Settings" panel are from the node marked by the red square.
2. **Defining workload -** workload is characterized after finishing the model, being configured through specific interfaces. Workloads can be deterministic, randomly defined or read from a trace file.
3. **Simulating the model -** simulation is queue-based, demanding the conversion from the iconic model to a queue model, including the load model. Service centers represent the defined computing nodes and communication channels. These centers attend events representing the computing tasks (workload) in the grid.
4. **Exhibiting results -** several plots and tables are available, showing results such as the load allocated to each node, average turnaround times, communication delays, system usage by each user. It also produces a new synthetic trace file, containing the load just simulated.



Machine icon configuration

Configuration for the icon#: 0

Properties	Values
Label	mac0
Owner	user1
Computing ...	10000.0
Load Factor	0.0
Cores	2
Primary Sto...	8.0
Secondary	2000.0

Properties

Local ID: 0
Global ID: 0
Label: mac0
X-coordinate: 136
Y-coordinate: 80
Computing power: 10000.0
Load Factor: 0.0
Master
Scheduling algorithm: WQR
Schedulable Nodes:

model opened

Figure 1. Modeling interface of iSPD.

3.2. Modeling Cloud Services

As indicated, iSPD is already capable of simulating several mechanisms involved with cloud computing, such as network traffic, computing loads, and scheduling policies. Therefore, its modification to simulate clouds deals mostly with the insertion of virtualization, resource provisioning and operational costs. These elements have different behavior for IaaS and PaaS, demanding different modifications in iSPD. In general, the new capabilities for these classes should include:

- **IaaS:** elements to model virtual machines (VM), virtual machine manager (VMM), VM resource allocation policies, utilization costs and resource availability;
- **PaaS:** elements to model Service Level Agreement (SLA) and a better description of the workload, including the definition of the applications that compose the workload.

3.3. Modeling Cloud Elements

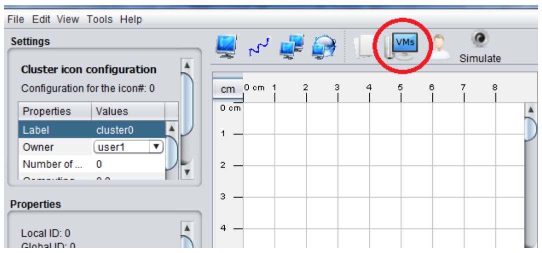
The main difference between cloud and grid computing is the definition of virtual machines. Since VMs are abstractions of real machines, they were modeled in an approach similar to the one used for single computational resources. To allow the definition of VMs a new icon, shown inside the red circle in Figure 2, was included.

Other entities used in grid models had to have their characteristics expanded in order to include information cloud's data. These entities and respective modifications, shown in boldface fonts, are described in the following list:

- Physical machine - defined by:
  - Identification of the machine's owner;
  - Number of processing cores;
  - Single core computing power (in MFlops);
  - Percentage of resources used by local/other processes;
  - **Main memory (in MB) and virtual memory (in GB)**, for VM configuration and allocation;
  - **Costs of CPU (in \$/core/hour), memory (in \$/Mb/hour) and storage (in \$/GB/hour)**;
  - **Node function**, defining if it hosts a VMM or is only a processing node.
- Cluster - modifications only in individual physical nodes as described above.
- Link and internet - followed the same structure used for grid simulations.

- Virtual Machine (VM) - defined by:
  - **User renting the VM**;
  - **VMM host**;
  - **Number of cores demanded**;
  - **Amount of main memory needed (in MB)**;
  - **Amount of secondary memory needed (in GB)**;
  - **Operating System installed in the VM (to simulate PaaS systems)**.

When either IaaS or PaaS simulation is chosen in the start window, the virtual machine icon is enabled, as shown in Figure 2. The parameters just presented can be provided to the cloud model through a set of different interfaces, specially built for cloud modeling. One of them is a new panel to model a host, shown in Figure 3, where different parameters must be provided to model a machine if the system is a grid 3a or a cloud 3b. The red box in 3b highlights VM related fields, such as costs and VMM policies.



Settings

Cluster icon configuration

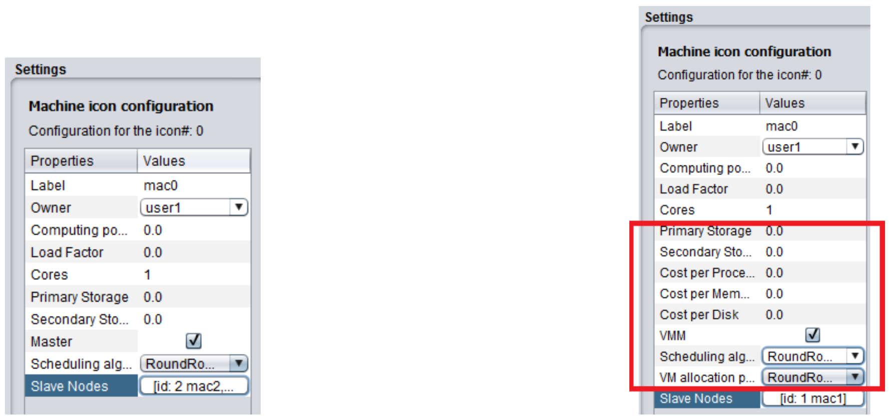
Configuration for the icon#: 0

Properties	Values
Label	cluster0
Owner	user1
Number of ...	0

Properties

Local ID: 0

Figure 2. Interface with an icon for VM modeling.



Settings

Machine icon configuration

Configuration for the icon#: 0

Properties	Values
Label	mac0
Owner	user1
Computing po...	0.0
Load Factor	0.0
Cores	1
Primary Storage	0.0
Secondary Sto...	0.0
Master	✓
Scheduling alg...	RoundRo...
Slave Nodes	[id: 2 mac2,...

(a) Grid configuration

Settings

Machine icon configuration

Configuration for the icon#: 0

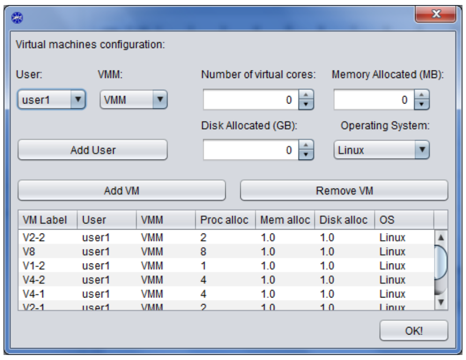
Properties	Values
Label	mac0
Owner	user1
Computing po...	0.0
Load Factor	0.0
Cores	1
Primary Storage	0.0
Secondary Sto...	0.0
Cost per Proce...	0.0
Cost per Mem...	0.0
Cost per Disk	0.0
VMM	✓
Scheduling alg...	RoundRo...
VM allocation p...	RoundRo...
Slave Nodes	[id: 1 mac1]

(b) Cloud configuration

Figure 3. Panels for host configuration in grids and clouds.

After modeling and configuring the physical nodes, it is necessary to model the virtual machines (VM). Figure 4 shows the window where this is accomplished. To model a VM the user has to provide values for its attributes, selecting which user is renting the VM and which VMM will be in charge of its management. The attributes of a given VM include demands for multicore processing, main memory and storage.

Finally, after all resources involved in the cloud have been modeled, it is necessary to model its workload. In iSPD a grid workload can be characterized following three approaches: deterministic, random, and by trace files. The same approaches can be extended


Virtual machines configuration:

User: user1
VMM: VMM
Number of virtual cores: 0
Memory Allocated (MB): 0
Disk Allocated (GB): 0
Operating System: Linux

VM Label	User	VMM	Proc alloc	Mem alloc	Disk alloc	OS
V2-2	user1	VMM	2	1.0	1.0	Linux
V8	user1	VMM	8	1.0	1.0	Linux
V1-2	user1	VMM	1	1.0	1.0	Linux
V4-2	user1	VMM	4	1.0	1.0	Linux
V4-1	user1	VMM	4	1.0	1.0	Linux
V2-1	user1	VMM	2	1.0	1.0	Linux

Figure 4. Window for VM configuration.

to cloud simulation if task grouping is added, since a given user can submit applications composed by one or several tasks, instead of the individual tasks managed in grids, even considering workflows. The new workload profile is shown in Figure 5.

In Figure 5 one can see that the user models applications instead of isolated tasks. Each application is configured through its internal tasks (either independent or workflow) and their common attributes. Attributes for the internal tasks (occurrence, demanded computation and communication) are the same either in grid or cloud simulation.

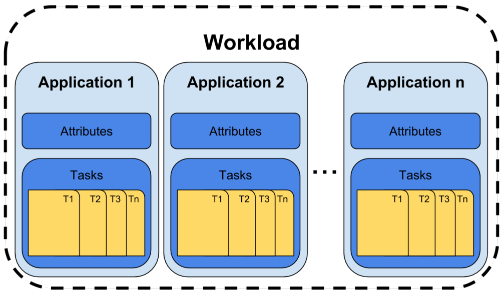


Figure 5. Workload characterization for IaaS clouds

3.4. Simulating Cloud Elements

The simulation engine had to be modified due to the inclusion of VMs and the modifications in some of the original elements. A new processing flow was defined, introducing steps to manipulate VMs. The creation of a VM is performed following the same approach used to create tasks in the system. This means inserting events for VM creation in the events list, with the needed information (resource requirements, etc). This event will be processed at the proper instant during simulation, triggering the initialization of usage counters for that VM.

When the events list becomes empty, i.e. all jobs are concluded, the simulation engine has to perform an additional step in order to finalize the VMs. In this step, iSPD walks through the physical hosts searching for active VMs. The VMs found are marked as “destroyed”, meaning that their usage counters will be stopped, allowing the derivation of utilization costs from these counters.

A final step in the simulation is the generation of performance metrics from that execution. They include all of the metrics already provided in the grid’s version of iSPD as well as metrics related to the allocation of VMs and costs of utilization, which are:

1. Allocation metrics, including the number of VMs actually allocated, VMs rejected due to insufficient resources, and VMs allocated per machine.
2. Usage Cost metrics, including partial and global costs calculated from the amount of resources used and for how long. The costs of processor, memory and disc usage are given by equations 1a, 1b and 1c, where "*proc. rate*" is the cost of using a specific physical machine for example, and "*alloc. time*" is the period of allocation in seconds.

$$\begin{aligned} \textit{Proc. cost} &= \textit{proc. rate} * (\textit{alloc. time}/3600) && \text{(1a)}\\ \textit{Mem. cost} &= \textit{memory rate} * (\textit{alloc. time}/3600) && \text{(1b)}\\ \textit{Disk cost} &= \textit{disk rate} * (\textit{alloc. time}/3600) && \text{(1c)} \end{aligned}$$

Finally, several output charts are produced by iSPD, providing enough information for either a cloud user or a cloud provider. For IaaS simulations the charts included are:

- General report about the simulation;
- Report about service and execution of tasks;
- Charts about the distribution of tasks among VMs;
- Charts about the distribution of VMs among physical machines.

4. Validation and Evaluation

In order to validate, and evaluate, iSPD's version for cloud simulation we considered two aspects. First we evaluated its accuracy through the comparison of results produced by iSPD against those produced by CloudSim and those measured in a real cloud. Second we evaluated how one could model and estimate the performance of a cloud environment under different VM allocation and scheduling policies.

4.1. Accuracy Tests

In order to evaluate the accuracy of iSPD we measured the performance of specific applications running on a real cloud and compared them to results achieved simulating a model of that cloud in iSPD and CloudSim. The real cloud was established through GCE (Google Compute Engine), which is an IaaS service provided by Google. This choice was made due the massive availability of GCE, although similar tests could be done with Amazon's EC2 or any other IaaS provider. The GCE cloud was assembled with three different configurations, given by:

- **g1-small:** one inexpensive virtual core, 1.7 GB of RAM and 10 GB of disk;
- **n1-standard-1:** one virtual core, 3.75 GB of RAM and 10 GB of disk;
- **n1-highcpu-2**: with two virtual cores, 1.8 GB of RAM and 10 GB of disk.

To perform the tests we considered a simple topology, where four computing nodes were linked to a front-end. Each host ran Ubuntu 14.04 LTS Trusty Tahr as the operating system. The workload applied to these hosts consisted in two python programs. One ran in the front-end and triggered four threads that running the second program in the other VMs. The second program was a server that repeated 20 million additions whenever requested, returning the time spent to perform this task.

This environment was modeled in both CloudSim and iSPD. Since both simulators use different units for computing power, it was necessary to produce an equivalence table for the VMs provided in GCE (Table 2). To produce such table we considered the average time to execute the program that ran in the servers, when running in a processor with measured speed of 51.2 Gflops, resulting in an average of 2.17s. This means that the server's code represents an average load of 111.104 Gflop.

Table 2. Equivalence table for computing power in the GCE machines.

System	Execution Time	Computing Power (Gflops)
g1-small	5.095s	21.805
n1-standard-1	2.419s	45.938
n1-highcpu-2	2.338s	47.518

An equivalent procedure, using the *traceroute* command, determinated the communication bandwidths available in the real cloud. The results are shown in Table 3.

Table 3. Equivalence table for communication bandwidths in the GCE cloud.

System	Transmission time (ms)	Bandwidth (Mbps)
g1-small	1.360	0.366
n1-standard-1	1.679	0.273
n1-highcpu-2	1.550	0.294

This environment was modeled in both simulators. The model in iSPD was easily created through its iconic interface, resulting in the model presented in Figure 6, where **node 0** is the front-end. For CloudSim we had to write a Java program, not presented here, with the classes and methods to model the system, both physical and virtual, including a method to generate random numbers.

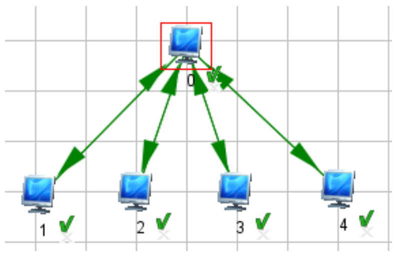
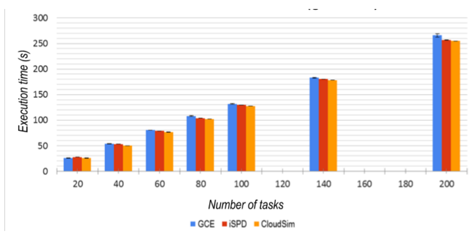
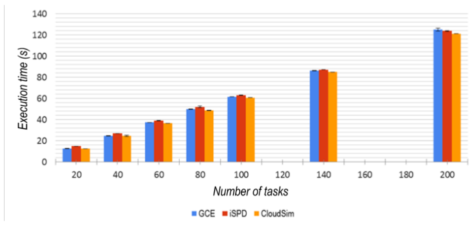


Figure 6. iSPD model for the GCE test cloud.

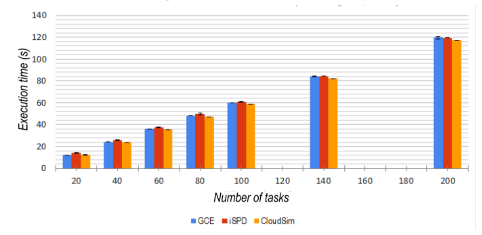
The results achieved are presented in the plots in Figures 7a, 7b and 7c, respectively for systems running *"g1-small"*, *"n1-standard-1"*, and *"n1-highcpu-2"* machines. Each system was evaluated using 20, 40, 60, 80, 100, 140, and 200 tasks, and the results shown are the average of 5 runs to achieve statistical convergence. Both simulators produced accurate results, with the simulated times from iSPD slightly higher than CloudSim's since the time spent to start a VM is accounted only in the iSPD's model.



(a) "g1-small" machines



(b) "n1-standard-1" machines



(c) "n1-highcpu-2" machines

Figure 7. Simulated and real times for GCE, iSPD and CloudSim

Table 4. Average times, in seconds, of real execution (GCE) and simulations (iSPD and CloudSim) for the *"n1-highcpu-2"* VMs.

tasks	GCE	iSPD	error (%)	CloudSim	error (%)
20	12.06	14.30	18.57	12.40	2,82
40	24.18	25.84	6.87	23.74	1,82
60	36.10	37.54	3.99	35.42	1,88
80	48.08	49.98	3.95	47.14	1,96
100	60.10	61.00	1.50	58.80	2,16
140	84.38	84.42	0.05	82.16	2,63
200	119.74	119.46	0.23	117.24	2,09

The average error was lower than 2%, being higher when less tasks were simulated (the highest one was 18% for 20 tasks in the *"n1-highcpu-2"* machines). The larger margin of error for smaller tests is also explained by the inclusion of the VM startup time, which has a larger impact for fewer tasks. Table 4 summarizes the numerical results for the *"n1-highcpu-2"* machines, including the relative errors for both simulators. In that table it is possible to observe that while iSPD was less accurate for small workloads, it had better performance for larger ones. Since it is reasonable to expect that no one will set a cloud to run just few tasks, its lower accuracy in such situation can be disregarded.

4.1.1. Evaluating simulator's performance

Although the major contribution introduced with iSPD is to facilitate modeling of clouds, it is also necessary to evaluate the performance of the simulation process. This is done measuring the time needed to simulate a specific cloud running different sets of tasks, comparing them with those needed by CloudSim. In this test we considered a larger system, with:

- 10 physical hosts, with two 50 Gflops cores each, 1GB of RAM and 10 GB of disk;

- 1 VMM, running the *first-fit* policy for VM allocation and *round-robin* for task scheduling;
- 20 VMs, each requiring 2 cores, 512 MB of RAM and 5 GB of disk.

The workload was composed by tasks demanding between 50 to 100 Gflop of processing, and 5 to 30 Mbytes of data transfers. The number of simulated tasks ranged from 10 to 100,000 tasks. This model was simulated in a PC running Ubuntu, with i5-4200U and 4 GB of RAM. The time spent in the simulations, in seconds, for both simulators are shown in Figure 8. CloudSim is faster for smaller loads, but becomes very slow after 10,000 tasks, since its execution time grows exponentially while iSPD's grows linearly.

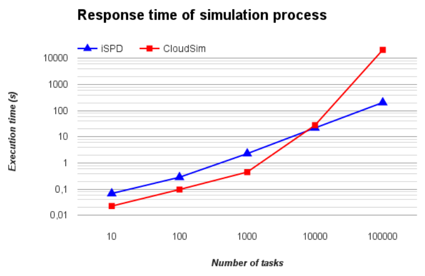


Figure 8. Response time to simulate the test cloud with iSPD and CloudSim.

4.2. Simulation of Allocation Policies

A second line of tests examined how iSPD can be used to evaluate the impact in cloud's performance of different VM allocation policies, and/or task scheduling policies. We used a model with 7 physical machines, one of them as the front-end hosting the VMM. Each physical machine was modeled with different amounts of 50 Gflops cores, 1 GB of RAM and 10 GB of disk. The machines were labeled following their number of cores, that is, they were the "8core", the "6core", the "4core", the "2core-1", the "2core-2", and the "1core" machines. This model is shown in Figure 9, from iSPD's modeling interface.

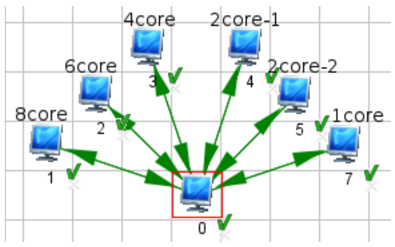


Figure 9. Cloud model used to evaluate allocation policies.

A total of 8 VMs were deployed to that system, using little memory and disk to make the allocation to depend only of the number of cores required. The modeled VMs were named according to the number of cores required.

The allocation policies evaluated were *round-robin* (RR), *first-fit* (FF), *first-fit decreasing* (FFD) and *volume*, all of them heuristic-based. For each one we repeated the allocation procedure 100 times, with the request order randomly generated each time. After each simulation we collected information about the average number of cores occupied, rejected VMs, and average occupation rates of the physical cores. Table 5 displays these metrics for the policies evaluated.

Table 5. Averages for VM rejection and core occupation for different VM allocation policies.

Allocation Policy	Rejected VMs	Cores occupied	Occupation rate (%)
RR	0.90	17.34	75
FF	0.89	18.62	81
FFD	0.00	23.00	100
Volume	1.00	15.73	68

It is possible to see that the *first-fit decreasing* heuristic has a better performance than the other ones. This occurs because FFD allocates the VMs in a decreasing order, that is, it allocates first the VM that demands more resources. This approach avoids the allocation of a larger physical machine to a smaller VM, what may compromise a latter allocation.

In Figure 10 one can see a chart produced by iSPD showing the number of VMs allocated to each of the physical nodes and the number of rejected allocations. In particular, the chart shown was produced using the "First-Fit" heuristic and, in that specific allocation, a VM was rejected because there was not sufficient resources to run it.

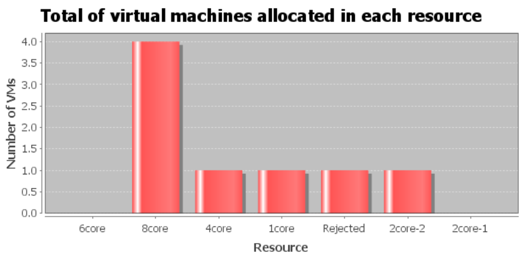


Figure 10. Allocation chart for the *first-fit* heuristic

5. Concluding Remarks

From the results presented here it is possible to conclude that iSPD's cloud extension is a good solution for cloud simulation. As described in the previous sections, the modifications introduced did not change the iconic-based nature of modeling with iSPD, that is, all the modeling work continue to be performed through graphical interfaces. Also, the simulation engine still produces statistically accurate results, and does this as fast as comparable simulators.

The tests showed that one can use iSPD either to evaluate simple execution times for a set of tasks in the cloud, as well as, to evaluate more complex relations, such as the impact of allocation policies. In any case the simulator was easy to use and provided charts easy to understand.

When compared to CloudSim, it was observed that one can model clouds much faster with iSPD, since there is no need to write Java codes. It is also possible to see that iSPD is faster than CloudSim when someone wants to simulate larger systems. The only drawback is its accuracy lower for small-sized workloads, which can be dismissed since these situations usually do not demand using a simulator as a performance analysis tool.

One future extension to this work will be the simulation of PaaS systems. From studies already conducted, PaaS simulation can be achieved without larger modifications to the current version of iSPD. This is a great advantage, since from our knowledge, there is no cloud simulator capable of PaaS simulation.

Availability

The iSPD can be downloaded from our github page at https://github.com/gspd/ispd-merge

Acknowledgment

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

References

Buyya, R., Broberg, J., and Goscinski, A. M. (2011). *Cloud Computing Principles and Paradigms*. Wiley Publishing.

Buyya, R., Ranjan, R., and Calheiros, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *CoRR*, abs/0907.4878.

Casanova, H. (2001). 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.

Castane, G. G., Nunez, A., and Carretero, J. (2012). iCanCloud: A brief architecture overview. In *Proc. of the 2012 IEEE 10th Intl Symp on Parallel and Distributed Processing with Applications*, ISPA'12, pages 853–854. IEEE Computer Society.

Chen, W. and Deelman, E. (2012). Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–8.

GridSim (2022). Gridsim's website. Available at <http://www.cloudbus.org/gridsim/>.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296.

Kliazovich, D., Bouvry, P., Audzevich, Y., and Khan, S. (2010). Greencloud: A packet-level simulator of energy-aware cloud computing data centers. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5.

Manacero, A., Lobato, R., Oliveira, P., Garcia, M., Guerra, A., Aoqui, V., Menezes, D., and Da Silva, D. (2012). iSPD: an iconic-based modeling simulator for distributed grids. In *Proc. of the 45th Annual Simulation Symposium*, pages 5:1–5:8. SCS.

Mell, P. M. and Grance, T. (2011). SP 800-145. The NIST definition of cloud computing. Technical report, National Institute of Standards & Technology, Gaithersburg, USA.