A Study on the Performance of Oracle Grid Engine for Computing Intensive Applications

Vladi Kolici
Polytechnic University of Tirana
Tirana, Albania
Email: vkolici@fti.edu.al

Albert Herrero
Universitat Politècnica de Catalunya
Barcelona, Spain
EMail: aherrero@lsi.upc.edu

Fatos Xhafa
Universitat Politècnica de Catalunya
Barcelona, Spain
EMail: fatos@lsi.upc.edu

Leonard Barolli
Fukuoka Institure of Technology
Fukuoka, Japan
Email: barolli@fit.ac.jp

***Abstract*—Computing intensive applications are an important family of applications in distributed computing domain. They have been object of study using different distributed computing paradigms and infrastructures. Such applications distinguish for their demanding needs for CPU computing, independently of the amount of data associated with the problem instance. Among computing intensive applications, there are applications based on simulations, aiming to maximize system resources for processing large computations for simulation. In this paper, we consider an application that simulates scheduling and resource allocation in a Grid computing system using Genetic Algorithms. In such application, a rather large number of simulations is needed to extract meaningful statistical results about the behaviour of the simulation results. We study the performance of Oracle Grid Engine for such application running in a Cluster of high computing capacities. Several scenarios were generated to measure the response time and queuing time under different workloads and number of nodes in the cluster.**

Keywords: Grid Computing, Cloud Computing, Simulation, Oracle Grid Engine, Computing Intensive Applications, Scheduling, Genetic Algorithms.

I. Introduction

Computing intensive applications (also known as compute intensive, computer intensive or computation intensive) are a family of applications arising in large simulations [5], [8] from many fields including bio-medicine [4] and genomics [9], finance [17], gaming, image processing [10], embedded applications, etc. that perform computationally intensive work and usually might need to run in batch mode for an extended period of time. These kind of applications are fuelled by Grid computing, Cloud computing, GPU computing and Multi-core computing paradigms. The study of such applications aims to shed light on the suitable computing paradigm as well as the underlying distributed computing infrastructure (HPC clusters, clouds, etc.) that achieves an efficient utilization of available computing resources while running hundreds to thousands of simultaneous jobs.

Among computing intensive applications, there are applications based on simulations, aiming to maximize system resources for processing large computations for simulation. Running efficient simulations in a distributed computing environment is a long standing problem in distributed computing. In this paper, we consider an application that simulates scheduling and resource allocation in a Grid computing system using Genetic Algorithms. In such application, a rather large number of simulations is needed to extract meaningful statistical results about the behaviour of the simulation results. We study the performance of Oracle Grid Engine for such application running in a HPC Cluster of high computing capacities. To that end, we consider running multiple simulations of the Grid scheduler based on Genetic Algorithms. We use several parameters to measure the efficiency of the HPC Cluster under Oracle Grid Engine. Thus, scheduling scenarios from small to large size comprising a variety of number of machines in the system and different number of tasks are considered. Additionally, we consider different degrees of dynamics of the Grid system to capture realistic features of dynamics of resources (such as resource failure) in Grid and Cloud computing systems. Several scenarios were generated to measure the response time and queuing time under different workloads and number of nodes in the cluster.

The rest of the paper is organized as follows. In Section II we briefly refer to different simulators in Grid and Cloud computing systems for scheduling and resource allocations. The intensive computing scenarios are presented in Section III, where we refer to the Grid simulator, scheduling using Genetic Algorithms under different scenarios of number of machines and number of tasks to be allocated. The results of the experimental study and their evaluation is presented

978-1-4799-6387-4/14 $31.00 © 2014 IEEE
DOI 10.1109/INCoS.2014.39

in Section IV. We end the paper in Section V with some conclusions and remarks for future work.

II. Simulations in Grid and Cloud computing systems

In this section, we briefly review some concepts and simulation tools in Grid and Cloud computing systems (refer to [6] for more details). As Grid computing is a precursor of Cloud computing, which uses the virtualization of the resources as a core feature, the simulators in Grid and Cloud computing share many features, especially those referring to simulating the underlying distributed infrastructures. As a matter of fact, most of Grid computing simulators can simulate features of Cloud computing. Indeed, many Grid simulators have been further developed to cover Cloud computing.

A. Grid simulators

SimGrid: SimGrid [3] is a C-based discrete event job scheduling simulation library which provides highly accurate network model for TCP and non-TCP transport.

GridSim: GridSim [1] is a Java based discrete event grid scheduling simulator built on top of SimJava which provides high extensibility and portability through Java and thread technologies.

HyperSim-G: HyperSim-G [14] extends HyperSim simulation package [12] which is an open source, general-purpose discrete event simulation library developed in C++. The main characteristics of HyperSim-G simulator are:

- Efficiency - obtains high performance simulations by modeling efficiently Grid environments.
- Scalability - scales very well to the burdens in the Grid size (both number of jobs and hosts) due to the dynamics of Grid systems because HyperSim-G, in contrast to other simulation packages, is not thread-based.
- Statistics - it offers extended statistical results on jobs, hosts and more generally about the influence of different scheduling policies as well as of other Grid parameters in the overall performance of the Grid system.
- HyperSim-G can be run in just a single run mode (one simulation is performed) which provide useful information about important parameters (makespan, flowtime, total potential time, total idle time, total busy time, etc.) or in many independent runs mode (many runs of the simulator are performed) the output results are averaged over the number of independent runs, showing also the standard deviation and confidence interval (95%).
- Parametrization - allows different types of Grid systems modeling and Grid scenarios arising in real Grid Computing systems.
- Local policies - Hosts in HyperSim-G can operate under different local scheduling policies (HyperSim-G currently supports the Space Share Policy).
- Scheduling algorithms- facilitates integration of different scheduling implementations. The simulator's design allows scheduling algorithms to be de-coupled from the simulator.
- Simulation traces - provides the full simulation trace by indicating a parameter for trace generation which is useful to understand the behavior of the simulator according to different parameter settings.
- Performing independent runs - allows to perform several independent runs, i.e. running the simulator by simply indicating the number of desired independent runs.

There is a web interface [16] for the HyperSim-G which facilitates running the simulator by remote users who can configure the simulator, chose scheduling mode, etc.

B. Cloud simulators

In a similar vein as for Grid computing, several Cloud computing simulators have been reported. Among them there are:

CloudSim: CloudSim [2] is a software framework for simulation, modeling and experimentation of Cloud Computing Infrastructures and applications. It follows the GridSim simulator for Grid computing. There are different variants available in CloudSim such as CloudAnalyst, Network CloudSim, EMUSIM and MDCSim.

iCanCloud: iCanCloud [11] is a simulation platform aimed to model and simulate cloud computing systems. The main objective of iCanCloud is to predict the trade-offs between cost and performance of a given set of applications executed in a specific hardware, and then provide to users useful information about the costs. However, iCanCloud can be used by a wide range of users, from basic active users to developers of large distributed applications.

GreenCloud: GreenCloud [7] is a simulation environment, for advanced energy-aware studies of cloud computing data centers developed as an extension of a packet-level network simulator Ns2. It offers a detailed fine-grained modeling of the energy consumed by the elements of the data center, such as servers, switches, and links.

Simulation Program for Elastic Cloud Infrastructure: SPECI [13] is a simulator of performance and behavior of large data centers. The implementation of SPECI can be divided in two groups. The first consists in data center layout and topology - contains classes for each type of component in the data centre, such as nodes and network links. These components mimic the operations of interest in the observed data centre, such as the transfer of network packets, maintaining subscriptions to other nodes, and keeping subscriptions up to date using the policy chosen for the experiment. The components have monitoring points that can be activated

as required by the experiment. The second is devoted to experiment execution and measuring - is built upon SimKit, which offers event scheduling as well as random distribution drawing.

III. Computing intensive scenarios using HyperSim-G simulator

In this section, we present the generation of several computing intensive scenarios used in this study using HyperSim-G simulator.

A. Scheduling

One common scheduling type in large scale distributed systems is that of independent tasks, in which tasks submitted to the system are independent and are not pre-emptive. This type of scheduling arises in real life applications where independent users submit their tasks or whole applications to the Grid system or in case of applications that can be split into independent tasks such as in parameter sweep applications, data mining and massive data processing, etc.

The problem is formally defined as: (a) a set of independent *jobs* that must be scheduled. Any job has to be processed entirely in unique resource; (b) a set of heterogeneous *machines* candidates to participate in the planning; (c) the *workload* (in millions of instructions) of each job; and, (e) the *computing capacity* of each machine (in *mips*).

Among the possible optimization criteria for the problem, the minimization of *makespan* (the time when the latest job finishes) and *flowtime* (the sum of finalization times of all the jobs) and the maximization of (average) utilization of Grid resources are defined.

B. Genetic Algorithms

Genetic Algorithms (GA) [15] have proved to be a good alternative for solving a wide variety of hard combinatorial optimization problems. GAs are a population-based approach where individuals (*chromosomes*) represent possible solutions, which are successively evaluated, selected, crossed, mutated and replaced by simulating the Darwinian evolution found in nature.

One important characteristic of GAs is the tendency of the population to converge to a fixed point where all individuals share almost the same genetic characteristics. If this convergence is accelerated, by means of the selection and replacement strategy, good solutions will be faster obtained. This characteristic is interesting for the job scheduling problem in Grid systems given that we might be interested to obtain a fast reduction in makespan value of schedule.

GAs are high level algorithms that integrate other methods and genetic operators, therefore, in order to implement it for the scheduling problem, we used a GA template (see Alg. 1) and designed the encodings, inner methods, operators and appropriate data structures (see Fig. 1 for the GA parameters).

Algorithm 1 Genetic Algorithm

Generate the initial population P^0 of size μ;
Evaluate P^0;
while not termination-condition **do**
  Select the parental pool T^t of size λ; $T^t := Select(P^t)$;
  Perform crossover procedure on pairs of individuals in T^t with probability p_c;
  $P_c^t := Cross(T^t)$;
  Perform mutation procedure on individuals in P_c^t with probability p_m;
  $P_m^t := Mutate(P_c^t)$;
  Evaluate P_m^t;
  Create a new population P^{t+1} of size μ from individuals in P^t and P_m^t;
  $P^{t+1} := Replace(P^t; P_m^t)$
  $t := t + 1$;
end while
return Best found individual as solution;

Genetic Algorithm parameters

Population size	100	
Intermediate population size	98	
Maximal time to spend	60	(integer) (min:1, max: 10000)
Optimization strategy	true	true - hierarchic false - simultaneous
Number of evolution steps	100	(integer)(min: 1, max: 20000)
Selection choice	SelectRandom	
Selection extra parameter	0.75	(double)(min: 0, max: 1)
Crossover probability	0.80	(double)(min: 0, max: 1)
Selection crossover operator	CrossOnePoint	
Crossover extra parameter	0.50	(double)(min: 0, max: 1)
Mutation probability	0.40	(double)(min: 0, max: 1)
Selection mutation operator	MutateMove	
Mutation extra parameter	0.60	(double)(min: 0, max: 1)
Replacement methods parameters		
Replace only if better	false	true - Replace if Better false - Steady State, Replace if Better, Elitist Generational
Replace generational	false	true - Simple Generational false - Steady State, Replace if Better, Elitist Generational
Initialization method		
Selection start choice	StartLJFRSJFR	

Run

Figure 1. Parameters of GA.

C. The Grid simulator

HyperSim-G is a discrete event-based simulator for Grid systems (the reader is referred to Xhafa et al. [14] for

details.) In HyperSim-G the scheduling algorithms are completely separated from the simulator, which needs not to know the implementation of the specific scheduling methods. This requirement regarding scheduling is achieved through a "refactoring" design and using new classes and methods, as shown in Figure 2.

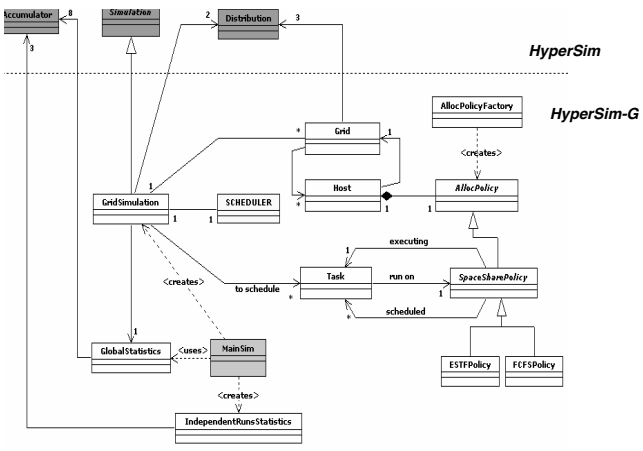

Figure 2. Scheduling entities in HyperSim-G.

Using the HyperSim-G simulator, we generated several computing intensive scenarios in this study. Indeed, the HyperSim-G is highly parameterizable: (1) size of the instance could be small, medium and large in terms of number of tasks and number of hosts; (2) scheduling type can be static and dynamic; (3) GA itself can be parametrized and (4) multiple independent runs can be executed (see Fig. 3).

```
Usage: sim [options]
Options:
  -n, --ttasks      <integer>    Total number of tasks (jobs)
  -b, --itasks      <integer>    Initial number of tasks
  -i, --iatime      <distrib>    Task mean interarrival time
  -w, --workload    <distrib>    Task mean work load (M.I.)
  -o, --ihosts      <integer>    Initial number of hosts
  -m, --mips        <distrib>    Host mean (M.I.P.S.)
  -a, --addhost     <distrib>    Add-host event distribution
  -d, --delhost     <distrib>    Del-host event distribution
  -f, --minhosts    <integer>    Minimum number of hosts
  -g, --maxhosts    <integer>    Maximum number of hosts
  -r, --reschedule               Reschedule unstart tasks
  -s, --strategy    <meta_p>     Scheduler meta policy
  -x, --hostselect  <host_p>     Host selection policy
  -y, --taskselect  <task_p>     Task selection policy
  -z, --activate    <wake_p>     Scheduler activation policy
  -l, --allocpolicy <local_p>    Host local scheduling policy
  -1, --nruns       <integer>    Number of runs
  -2, --seed        <integer>    Random seed
  -t, --trace       <filename>   Enable trace on output file
  -h, --help                     Shows this help
```

Figure 3. Parameters of HyperSim-G.

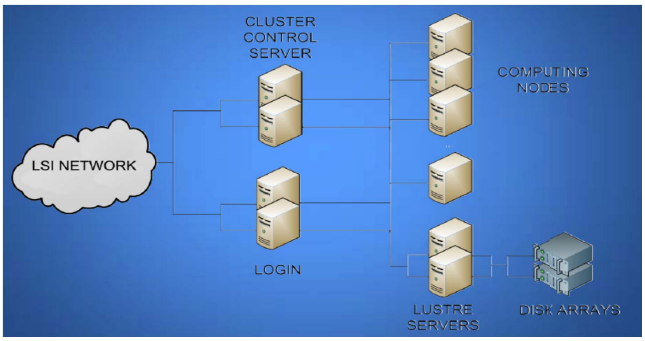

Figure 4. RDLab Cluster architecture

IV. Experimental Study

In this section, we present the design and results of the experimental study.

A. Experimental setup

We study the performance of Oracle Grid Engine in our HPC Cluster through two scenarios: 1) varying the size of the problem instance through the configuration of the HyperSim-G simulator and 2) varying the number of nodes used for computation. In the former, we used a fixed number of 16 nodes under static and dynamic configuration of the simulator. For the later, we fixed the size of the simulator instances to medium size (M) and varied the number of cluster nodes to 2, 4, 8, 16 and 32 nodes. In both cases, we measured the queuing time and response time.

B. RDLab Cluster and Oracle Grid Engine

The experimental study was carried out in the RDLab Cluster[1]. The RDlab belongs to the PlanetLab Europe project. The HPC infrastructure contains More than 70 nodes, 600 execution cores, 1 TByte of RAM, 10 TBytes of disk space and is used by about 120 users. The cluster uses Oracle Grid Engine[2] queuing system.

C. Computational results

1) First experiment: varying the computation load: For the first experiment, we used 16 nodes of the Cluster and the HyperSim-G configuration: static case (see Table I) and dynamic case (see Table II).

Under these configuration, we run simultaneously 2, 4, 8, 16 and 32 simulations. The graphical representation of queuing time and response time for the static case can be seen in Fig. 5 and Fig. 6 and for the dynamic case in Fig. 7 and Fig. 8 (in the figures, #JOBS refers to number of simulations–submitted JOBS to the qyueuing system).

As can be seen from the Fig(s) 5 to Fig. 8, both queuing time and consecutively the response time remain small and

[1]http://rdlab.lsi.upc.edu/index.php/en/
[2]http://www.sun.com/software/sge/

Table I
HYPERSIM-H SIMULATOR STATIC CONFIGURATION

	Small	Medium	Large
Init. hosts	32	64	128
Max. hosts	37	70	135
Min. hosts	27	58	121
MIPS	-	N(1000, 175)	-
Add host	-	C(9999999999)	-
Delete host	-	C(9999999999)	-
Total tasks	512	1024	2048
Init. tasks	384	768	1536
Workload	-	N(250000000, 43750000)	-
Interarrival	-	C(9999999999)	-
Activation	-	C(9999999999)	-
Reschedule	-	True	-
Host select	-	All	-
Task select	-	All	-
Number of runs	-	5	-

Table II
HYPERSIM-H SIMULATOR DYNAMIC CONFIGURATION

	Small	Medium	Large
Init. hosts	32	64	128
Max. hosts	37	70	135
Min. hosts	27	58	121
MIPS	-	N(1000, 175)	-
Add host	N(625000, 93750)	N(562500, 84375)	N(500000, 75000)
Delete host	-	N(625 000,93 750)	-
Total tasks	512	1024	2048
Init. tasks	384	768	1536
Workload	-	N(250000000, 43750000)	-
Interarrival	E(7812.5)	E(3906.25)	E(1953.125)
Activation	-	C(250000)	-
Reschedule	-	True	-
Host select	-	All	-
Task select	-	All	-
Number of runs	-	5	-

"resonable" for small and medium size simulation instances; however, they increase exponentially if more than 16 large size simulations are simultaneously run.

2) Second experiment: varying the number of nodes: In the second experiment, we fixed the size of the simulator instances to medium size (M) and varied the number of cluster nodes to 2, 4, 8, 16 and 32 nodes. The simulator configuration is shown in Table III. Under this configuration, the graphical representation of queuing time and response time can be seen in Fig. 9 and Fig. 10. As can be observed from these figures, a significant reduction in queuing time and response time was achieved by doubling the number of nodes from 2 to 32 nodes.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the results of a study on the performance of Oracle Grid Engine for computing intensive applications. Such applications arise in many fields and distinguish for their demanding needs for CPU computing, independently of the amount of data associated

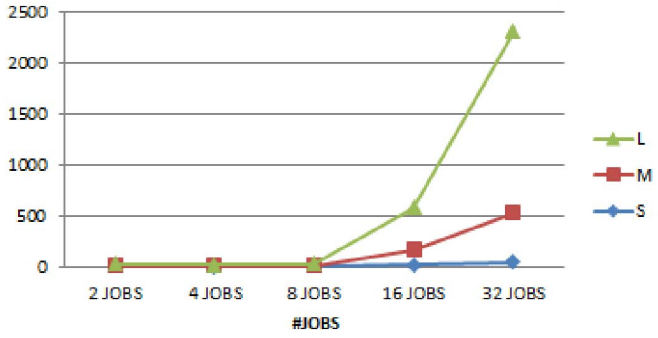

Figure 5. Queuing time (in *seconds*) for Small (S), Medium (M) and Large (L) size simulator instances for *static* configuration of simulator

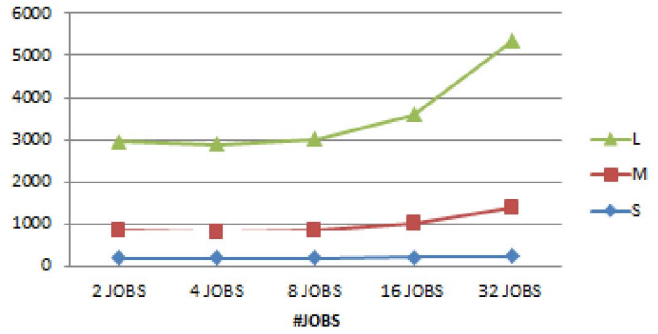

Figure 6. Response time (in *seconds*) for Small (S), Medium (M) and Large (L) size simulator instances for *static* configuration of simulator

with the problem instance. For the study, we considered an application that simulates scheduling and resource allocation in a Grid computing system using Genetic Algorithms. In such application, a rather large number of simulations is needed to extract meaningful statistical results about the behaviour of the simulation results. The experimental study was conducted in a Cluster of high computing capacities.

Figure 7. Queuing time (in *seconds*) for Small (S), Medium (M) and Large (L) size simulator instances for *dynamic* configuration of simulator

Figure 8. Response time (in *seconds*) for Small (S), Medium (M) and Large (L) size simulator instances for *dynamic* configuration of simulator

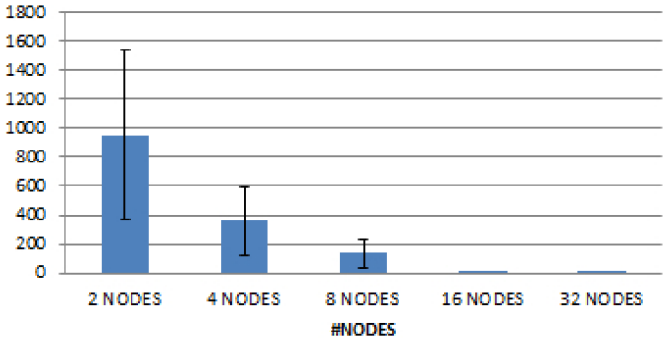

Figure 9. Queuing time (in *seconds*) for Medium (M) size simulator instances for different nodes in the cluster

Table III
HYPERSIM-H SIMULATOR CONFIGURATION USED IN SECOND EXPERIMENT

	Medium
Init. hosts	64
Max. hosts	70
Min. hosts	58
MIPS	N(1000, 175)
Add host	N(562500, 84375)
Delete host	N(625 000,93 750)
Total tasks	1024
Init. tasks	768
Workload	N(250000000, 43750000)
Interarrival	E(3906.25)
Activation	C(250000)
Reschedule	True
Host select	All
Task select	All
Number of runs	5

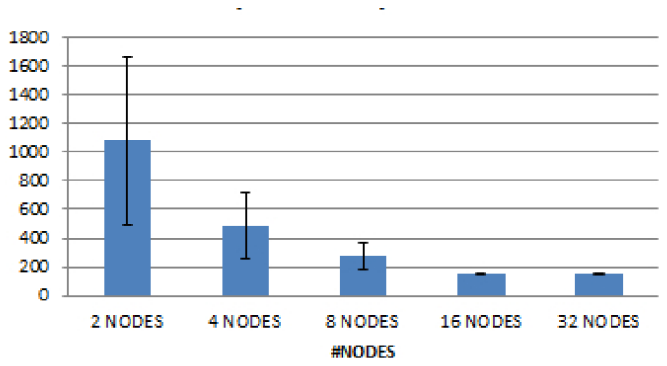

Figure 10. Response time (in *seconds*) for Medium (M) size simulator instances for different nodes in the cluster

Several scenarios were generated to measure the response time and queuing time under different workloads and number of nodes in the cluster.

In our future work we would like to make a study on the performance of Oracle Grid Engine using data intensive computing to measure specifically I/O operations and applications that are both computing intensive and data intensive.

REFERENCES

[1] R. Buyya and M. Murshed. Gridsim: A toolkit for the-modeling and simulation of distributed resource managementand scheduling for grid computing, Concurrency and Com- putation: Practice and Experience , vol. 14, no. 13-15, pp.11751220, 2002.

[2] R. Buyya, R. Ranjan, and R. N. Calheiros. Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities, International Conference on High Performance Computing & Simulation, HPCS '09, Pag. 1-11, 2009.

[3] H. Casanova. Simgrid: A toolkit for the simulation of application scheduling in Proceedings of First IEEE/ACM International Symposium on Cluster Computing and the Grid, Brisbane, Qld. , Australia , 2001.

[4] S. Clémençon, A. Cousien, M. Dávila Felipe, V. Chi Tran. On Computer-Intensive Simulation and Estimation Methods for Rare Event Analysis in Epidemic Models, 2013 http://arxiv.org/abs/1308.5830

[5] M.J. Fortin, G.M. Jacquez, B. Shipley. Computer-intensive methods Volume 1, pp 399402 in Encyclopedia of Environmetrics. Edited by Abdel H. El-Shaarawi and Walter W. Piegorsch John Wiley & Sons, Ltd, Chichester, 2002

[6] K. Goga, O. Terzo, P. Ruiu and F. Xhafa. Simulation, Modeling and Performance Evaluation Tools for Cloud Applications. In Proceedings of the 8th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS-2014, Birmingham, UK, 2-5 July 2014. IEEE CPS, *To appear*

[7] D. Kliazovich, P. Bouvry, Y. Audzevich, S. U. Khan. GreenCloud: A Packet-level Simulator of Energy-aware Cloud ComputingData Centers, IEEE Globecom 2010 proceedings, Page(s): 1 - 5, IEEE.

[8] Q. Liu and G. Wainer. Exploring Multi-Grained Parallelism in Compute-Intensive DEVS Simulations, In IEEE Workshop on Principles of Advanced and Distributed Simulation (PADS), 17-19 May 2010, Page(s): 1 - 8, Atlanta, GA, 10.1109/PADS.2010.5471652, IEEE

[9] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly and M.A. DePristo. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 2010 Sep;20(9):1297-303. doi: 10.1101/gr.107524.110

[10] A. Niedzicka. Computation-intensive image processing algorithm parallelization on multiple hardware architectures. In PARELEC '02. Proceedings. International Conference on Parallel Computing in Electrical Engineering, 2002, Page(s): 446 - 448, 10.1109/PCEE.2002.1115341, IEEE.

[11] A. Nunez, J.L. Vazquez-Poletti, A.C. Caminero, G.G. Castane, J. Carretero and I.M. Llorente. iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator. Journal of Grid Computing March 2012, Volume 10, Issue 1, pp 185-209

[12] S. Phatanapherom, P. Uthayopas, V. Kachitvichyanukul. Fast simulation model for grid scheduling using HyperSim, Simulation Conference, 2003. Proceedings of the 2003 Winter (Volume:2), Pag. 1494 - 1500, 2003.

[13] I. Sriram SPECI, a simulation tool exploring cloud scale data centres, CloudCom 2009, LNCS 5931, pp. 381-392, 2009, M.G. Jaatun, G. Zhao, and C. Rong (Eds.),Springer-Verlag Berlin Heidelberg 2009

[14] F. Xhafa, J. Carretero, L. Barolli and A. Durresi. Requirements for an Event-Based Simulation Package for Grid Systems. Journal of Interconnection Networks, Vol. 8, No. 2, 163–178, 2007, World Scientific Pub.

[15] F. Xhafa, J. Carretero, A. Abraham. Genetic Algorithm Based Schedulers for Grid Computing Systems. International Journal of Innovative Computing, Information and Control, Vol. 3, No.5, pp. 1-19, 2007.

[16] F.Xhafa, L. Barolli, D. Martos. A WEB Interface for Hypersim-G Grid Simulation Package, Proceedings of iiWAS, page 312-317, 2008.

[17] L. Yao, F.A. Rabhi and M. Peat. A Case Study in Using ADAGE for Compute-Intensive Financial Analysis Processes. Enterprise Applications and Services in the Finance Industry Lecture Notes in Business Information Processing, Volume 135, 2013, pp 91-111