Invited Paper

J Inf Process Syst, Vol.10, No.4, pp.491~502, December 2014
http://dx.doi.org/10.3745/JIPS.01.0004

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

On the Performance of Oracle Grid Engine Queuing System for Computing Intensive Applications

Vladi Kolici*, Albert Herrero**, and Fatos Xhafa**

Abstract—In this paper we present some research results on computing intensive applications using modern high performance architectures and from the perspective of high computational needs. 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 research work, 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 behavior 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—Benchmarking, Cloud Computing, Computing Intensive Applications, Genetic Algorithms, Grid Computing, Oracle Grid Engine, Scheduling, Simulation

1. INTRODUCTION

Computing intensive applications (also known as compute intensive, computer intensive or computation intensive) are a family of applications arising in large simulations [1,2] from many fields including bio-medicine [3] and genomics [4], finance [5], gaming, image processing [6], 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

Manuscript received October 14, 2014; accepted November 20, 2014.
Corresponding Author: Fatos Xhafa (fatos@lsi.upc.edu)
* Department of Electronic and Telecommunication, Polytechnic University of Tirana, Tirana, Albania. (kolici@fti.edu.al)
** Department of Computer Science, Universitat Politècnica de Catalunya, Barcelona 08034, Spain. (aherrero@lsi.upc.edu, fatos@lsi.upc.edu)

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 (GAs). In such application, a rather large number of simulations is needed to extract meaningful statistical results about the behavior 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 GAs. 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 remainder of this paper is organized as follows. In Section 2 we overview different simulators in Grid and Cloud computing systems for scheduling and resource allocations. The intensive computing scenarios are presented in Section 3, where we refer to the Grid simulator, scheduling using GAs under different scenarios of number of machines and number of tasks to be allocated. The results of the experimental study and their evaluation are presented in Section 4. We end the paper in Section 5 with some conclusions, lessons learned and remarks for future work.

2. SIMULATIONS IN GRID AND CLOUD COMPUTING SYSTEMS

In this section, we overview some concepts and simulation tools in Grid and Cloud computing systems (refer to [7] and [8] 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.

2.1 Grid Simulators

Several Grid simulators have been proposed in the literature. We discuss them next.

SimGrid

SimGrid [9] is a C-based discrete event job scheduling simulation library which provides highly accurate network model for TCP and non-TCP transport.

GridSim

GridSim [10] 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 [11] 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.
- *Execution modes* - 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 (at 95% confidence value).
- *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 [13] for the HyperSim-G which facilitates running the simulator by remote users who can configure the simulator, chose scheduling mode, etc.

2.2 Cloud Simulators

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

CloudSim

CloudSim [14] 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 [15] 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 [16] 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)

SPECI [17] is a simulator of performance and behavior of large data centers. The implementation of SPECI can be divided in two groups.

1. The first consists in data center layout and topology - contains classes for each type of component in the data center, such as nodes and network links. These components mimic the operations of interest in the observed data center, 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.
2. The second is devoted to experiment execution and measuring - is built upon SimKit, which offers event scheduling as well as random distribution drawing.

3. 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.

3.1 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,
d) 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.

3.2 Genetic Algorithms

GA [18] 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.

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;

Fig. 1. Genetic algorithm template.

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 Algorithm 1 in Fig. 1) and designed the encodings, inner methods, operators and appropriate data structures (see also Fig. 2 for the GA parameters).

3.3 The Grid Simulator

HyperSim-G is a discrete event-based simulator for Grid systems (the reader is referred to Xhafa et al. [11,13] 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 Fig. 3.

Using the HyperSim-G simulator, we generated several computing intensive scenarios in this study. Indeed, the HyperSim-G is highly parameterizable:

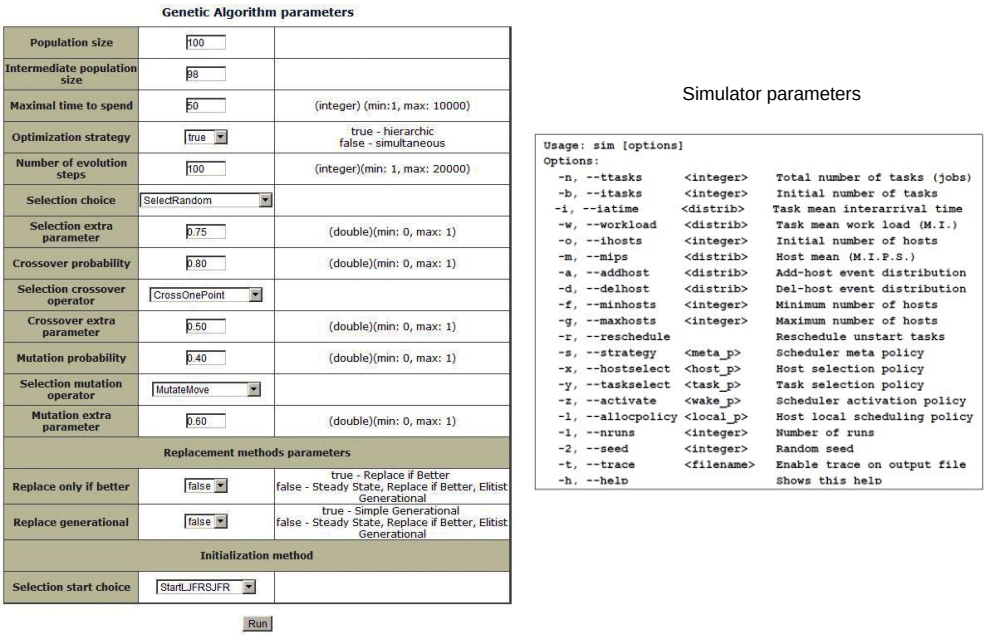
(1) size of the instance could be small, medium and large in terms of number of tasks and

number of hosts;

(2) scheduling type can be static and dynamic;

(3) GA itself can be parameterized; and

(4) multiple independent runs can be executed (see Fig. 2(b)).



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

Simulator parameters

```
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
```

(a) (b)

Fig. 2. Genetic algorithm parameters (a) and simulator parameters (b).

4. EXPERIMENTAL STUDY

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

4.1 Experimental Setup

We study the performance of Oracle Grid Engine in our HPC cluster (see Fig. 4) 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.

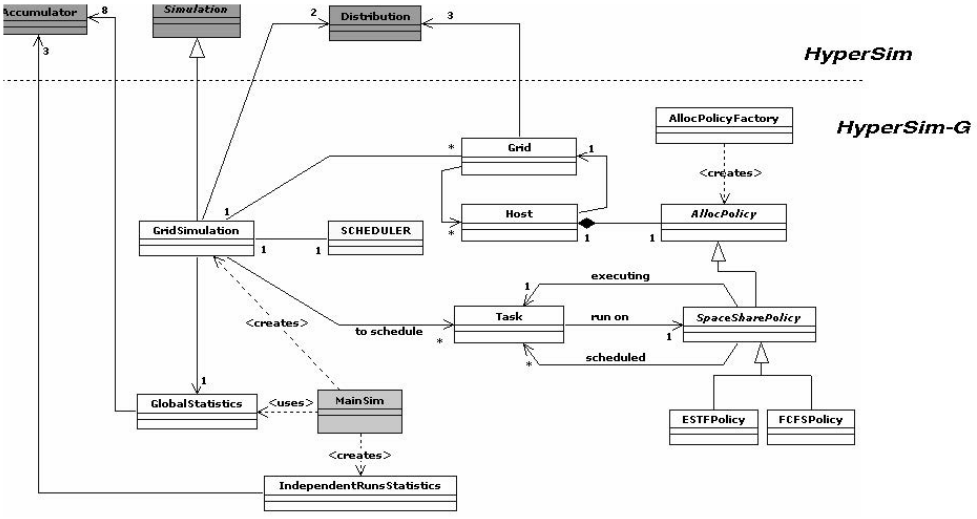


Fig. 3. UML design of the HyperSim-G simulator.

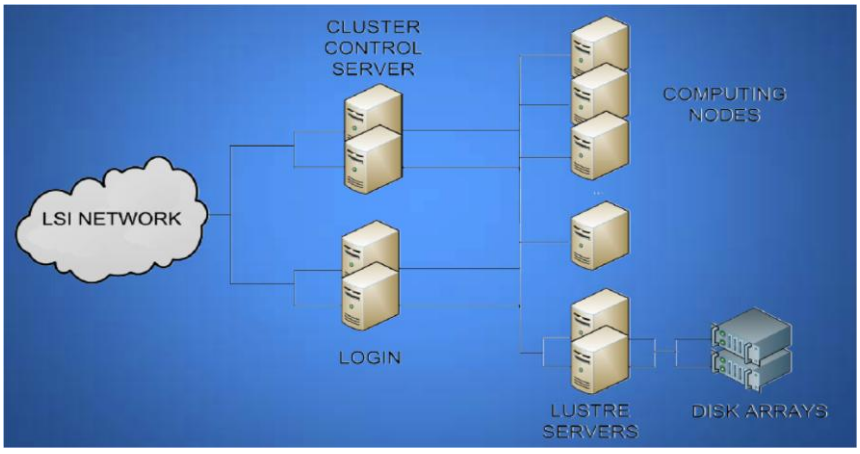


Fig. 4. HPC Cluster at the Research and Development Laboratory (RDlab), BarcelonaTech.

4.2 RDlab Cluster and Oracle Grid Engine

The experimental study was carried out in the Research and Development Laboratory (RDlab) Cluster (http://rdlab.lsi.upc.edu/index.php/en/).

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 (http://www.sun.com/software/sge/) queuing system.

Table 1. HyperSim-G simulator setup

Table 1a. 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 1b. 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	-

4.3 Computational Results

4.3.1 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 1a) and dynamic case (see Table 1b). Under these configurations, we run simultaneously 2, 4, 8, 16, and 32 simulations.

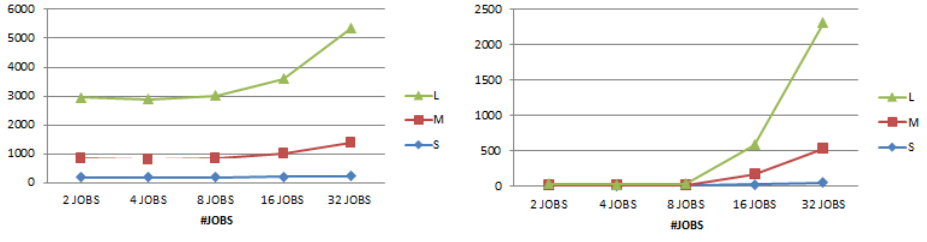
The graphical representation of queuing time and response time for the static case can be seen in Fig. 5(a) and (b) and for the dynamic case in Fig. 6(a) and (b) (in the figures, #JOBS refers to number of simulations—submitted JOBS to the queuing system).

As can be seen from the Figs. 5 and 6, both queuing time and consecutively the response time remain small and 'reasonable' for small and medium size simulation instances; however, they increase exponentially if more than 16 large size simulations are simultaneously run.

4.3.2 Second experiment: varying the number of nodes

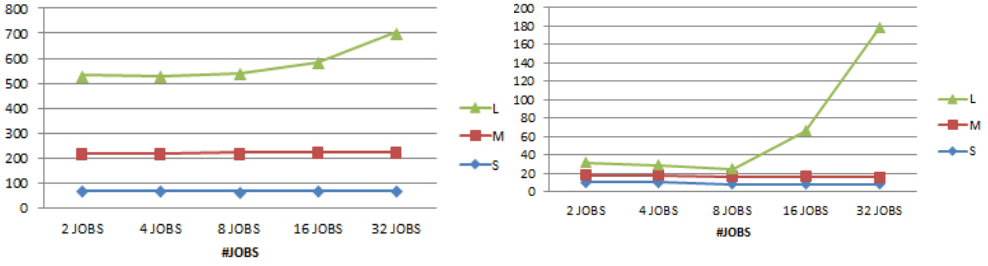
In the second experiment, we fixed the size of the simulator instances to medium size (M) and varied the number of cluster nodes to 2, 4, 8, 16, and 32 nodes. The simulator configuration is shown in Table 2.

Under this configuration, the graphical representation of response time and queuing time can be seen in Fig. 7(a) and (b), respectively. 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.



(a) (b)

Fig. 5. Results (in seconds) for static configuration: response time (a) and queuing time (b) for small (S), medium (M) and large (L) size simulator instances.



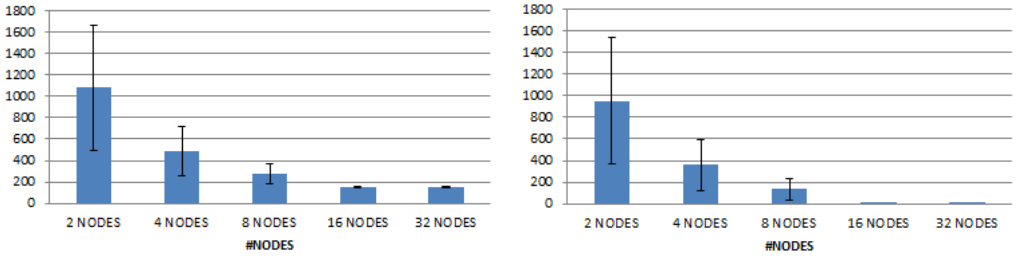
(a) (b)

Fig. 6. Results (in seconds) for dynamic configuration: response time (a) and queuing time (b) for small (S), medium (M) and large (L) size simulator instances.

Table 2. Simulator configuration parameters for the second experiment (medium size instances)

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



(a) (b)

Fig. 7. Results (in seconds) for medium size instances: response time (a) and queuing time (b).

5. 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 with the problem instance. For the study, we considered an application that simulates scheduling and resource allocation in a Grid computing system using GAs. In such application, a rather large number of simulations are needed to extract meaningful statistical results about the behavior of the simulation results. The experimental study was conducted 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.

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] M. J. Fortin, G. M. Jacquez, and B. Shipley, "Computer intensive sampling methods in ecology," in *Encyclopedia of Environmetrics*. Chichester: Wiley, 2002, pp. 399-402.

[2] Q. Liu and G. Wainer, "Exploring multi-grained parallelism in compute-intensive DEVS simulations," in *Proceedings of the 2010 IEEE Workshop on Principles of Advanced and Distributed Simulation*, Atlanta, GA, 2010, pp. 1-8.

[3] S. Clémençon, A. Cousien, M. D. Felipe, and V. C. Tran, "On computer-intensive simulation and estimation methods for rare event analysis in epidemic models," Aug. 2013; http://arxiv.org/pdf/1308.5830v1.pdf.

[4] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, ... and M. A. DePristo, "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data," *Genome Research*, vol. 20, no. 9, pp. 1297-1303, 2010.

[5] L. Yao, F. A. Rabhi, and M. Peat, "A case study in using ADAGE for compute-intensive financial analysis processes," in *Proceedings of the 6th International Workshop on Enterprise Applications and Services in the Finance Industry*, Barcelona, Spain, 2013, pp. 91-111.

[6] A. Niedzicka, "Computation-intensive image processing algorithm parallelization on multiple hardware architectures," in *Proceedings of International Conference on Parallel Computing in Electrical Engineering (PARELEC2002)*, Warsaw, Poland, 2002, pp. 446-448.

[7] 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 (CISIS2014)*, Birmingham, UK, 2014, pp. 226-232.

[8] V. Kolici, F. Xhafa, A. Herrero, and L. Barolli, "A study on the performance of Oracle Grid engine for computing intensive applications," in *Proceedings of the 6th International Conference on Intelligent Networking and Collaborative Systems (INCoS2014)*, Salerno, Italy, 2014, pp. 282-288.

[9] H. Casanova, "Simgrid: a toolkit for the simulation of application scheduling," in *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid*, Brisbane, Australia, 2001, pp. 430-437.

[10] 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: Practice and Experience*, vol. 14, no. 13-15, pp. 1175-1220, 2002.

[11] 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, pp. 163-178, 2007.

[12] S. Phatanapherom, P. Uthayopas, and V. Kachitvichyanukul, "Dynamic scheduling II: fast simulation

model for grid scheduling using HyperSim," in *Proceedings of the 35th Conference on Winter Simulation: Driving Innovation (WSC2003)*, New Orleans, LO, 2003, pp. 1494-1500.

[13] F. Xhafa, L. Barolli, and D. Martos, "A web interface for the HyperSim-G Grid simulation package," in *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services (iiWAS2008)*, Linz, Austria, 2008, pp. 312-317.

[14] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: challenges and opportunities," in *Proceedings of the International Conference on High Performance Computing & Simulation (HPCS'09)*, Leipzig, Germany, 2009, pp. 1-11.

[15] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "iCanCloud: a flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185-209, 2012.

[16] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *in Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2010***)**, Miami, FL, 2010, pp. 1-5.

[17] I. Sriram, "SPECI, a simulation tool exploring cloud-scale data centres," in *Proceedings of the 1st International Conference on Cloud Computing*, Beijing, China, 2009, pp. 381-392.

[18] J. Carretero, F. Xhafa, and A. Abraham, "Genetic algorithm based schedulers for grid computing systems," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 6, pp. 1-19, 2007.

Vladi Kolici

Vladi Kolici received his B.S. and M.S. in Telecommunication Engineering from the Polytechnic University of Tirana (PUT) in 1997 and 2005, respectively. He obtained his Ph.D. from PUT in May 2009. From 1997 to 2004, he was a Research Associate, from 2005 to 2011 he has been a Lecturer and at present he is Associate Professor at the Department of Electronics and Telecommunications, Faculty of Information Technology, PUT. He is teaching several courses in the areas of wireless and mobile networking, P2P systems and quality of services. His has published several papers in international and national conference proceedings in the areas of P2P and ad hoc networks. His research interests include P2P networks, wireless and mobile networks and high speed networks.

Albert Herrero

He received a M.Sc. degree in Software Engineering from Technical University of Catalonia, Spain in 2013. His research interest includes the areas of Web computing, High Performance Computing, Distributed Programming and Cloud Computing.

Fatos Xhafa

He holds a Ph.D. in Computer Science from the Department of Computer Science of the Technical University of Catalonia (UPC), Barcelona, Spain, where he holds a permanent position of Professor Titular. He was a Visiting Professor at Birkbeck College, University of London (UK) during academic year 2009-2010 and Research Associate at Drexel University, Philadelphia (USA) during academic term 2004/2005. Prof. Xhafa has widely published in peer reviewed international journals, conferences/workshops, book chapters and edited books and proceedings in the field. Prof. Xhafa has an extensive editorial and reviewing service and is actively participating in the organization of several international conferences. His research interests include parallel and distributed algorithms, security, optimization, networking and distributed computing. More information can be found at http://www.lsi.upc.edu/~fatos/.