2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing

A Web Interface for Meta-Heuristics Based Grid Schedulers

Fatos Xhafa
Department of Languages and Informatics Systems
Technical University of Catalonia, Spain
Email: fatos@lsi.upc.edu

Joanna Kołodziej
Department of Mathematics and Computer Science
University of Bielsko-Biała, Poland
Email: jkolodziej@ath.bielsko.pl

Marcin Bogdański
Faculty of Mechanical Engineering and Computer Science
University of Bielsko-Biała, Poland
Email: marcin.bogdanski@gmail.com

***Abstract*—The use of meta-heuristics for designing efficient Grid schedulers is currently a common approach. One issue related to Grid based schedulers is their evaluation under different Grid configurations, such as dynamics of tasks and machines, task arrival, scheduling policies, etc. In this paper we present a web application that interfaces the final user with several meta-heuristics based Grid schedulers. The application interface facilities for each user the remote evaluation of the different heuristics, the configuration of the schedulers as well as the configuration of the Grid simulator under which the schedulers are run. The simulation results and traces are graphically represented and stored at the server and can retrieved in different formats such as spreadsheet form or pdf files. Historical executions are as well kept enabling a full study of use cases for different types of Grid schedulers. Thus, through this application the user can extract useful knowledge about the behavior of different schedulers by simulating realistic conditions of Grid system without needing to install and configure any specific software.**

***Keywords*-Web interface, Grid scheduling, Simulation, Meta-heuristic.**

I. Introduction

Grid computing has emerged as a wide area distributed platform for solving the large-scale problems in science, engineering, etc. Computational Grid involves the combination of many computing resources into a network for the execution of computational tasks. The resources are distributed across multiple organizations, administrative domains having their own access, usage policies and local schedulers. The tasks scheduling and the effective management of the resources in such systems are complex and therefore, demands sophisticated tools for analyzing the algorithms performances before applying them to the real systems.

Simulation seems to be the most suitable way to analyze scheduling algorithms in large-scale distributed dynamic systems like Grid environment. It simplify the study of schedulers performances and avoid the overhead of coordination of the resources, which usually happens in the real-life Grid scenarios. Simulation is also effective in working with very large problems that require involvement of a large number of active users and resources, which is usually very hard to coordinate and build for real-life approaches. Moreover, in order to make useful the analysis of the performance and parameters tuning of different scheduling algorithms, a considerable number of independent runs is needed to ensure significant statistical results, which can be easily realized with Grid simulator.

Despite the work on the Grid simulation packages, there are still few research works on development of full-featured interfaces for Grid simulation packages. As a matter of fact, current Grid simulation packages are run through command line. In this work we propose a user-friendly Web interface to remote users and benefit from easy use features of the interface for running the HyperSim-G simulator. The proposed Web application, namely WEBGRID[1], facilitates the evaluation of the different scheduling heuristics, the configuration of the schedulers as well as the configuration of the Grid simulator, under which the schedulers are run. The simulation results and traces are graphically represented and stored at the server and can retrieved in different formats such as spreadsheet form or pdf files. Historical executions are stored in the system, which enables a full study of use cases for different types of Grid schedulers.

The rest of the paper is organized as follows. In Section II we highlight the Grid scheduling problems types and selected resolution methods. The HyperSim-G simulator as well as its integration with various schedulers are briefly described in Section III. In Section IV we present the main aspects of the Web application architecture and show its application in solving a specified instance of Grid scheduling problem. Finally, we conclude this work in Section V.

II. Scheduling in Grid Systems

A. Scheduling Problems

The purpose of the schedulers is to efficiently and optimally allocate tasks originated by applications to a set of available resources. In general, both tasks and resources could be dynamically added/dropped to/from the system. Scheduling in Grids remains a challenging NP-complete global optimization problem because of the large-scale heterogeneous structure of the system, and the existence of

[1]The application is available at the following Web page: http://weboptserv.lsi.upc.edu/WEBGRID/

local job dispatchers and resource owners geographically dispersed in different autonomous administrative domains.

The type of the scheduling problem in Grids is specified by the setting up the main scheduling attributes presented in Table I.

One of the most useful version of the scheduling problem is the Independent Job Scheduling, in which tasks are processed in batch mode. The main characteristics of this kind of scheduling in distributed systems is the requirement over tasks, arranged in batches, to be executed independently on the resources. Independent scheduling is very suitable to address in Grid systems, which are parallel in nature. In particular, independent batch especially in the case of the verification of the security assurance condition. The absence of dependencies among tasks makes it easier to pre-empty or re-schedule tasks. The independent scheduling problem formulation is usually based on the Expected Time to Compute (ETC) matrix model, in which an instance is defined by the following input data:

- a number of tasks to be scheduled (size of the batch)
- a number of resources
- the workload of tasks
- the computing capacity of resources
- the ready times indicating when machines will have finished the previously assigned tasks.
- the ETC matrix of size $nb\_tasks \times nb\_machines$, where $ETC[j][m]$ is the value of the expected time to compute task j in machine m.

The detailed destription of the types of scheduling problems in Grids can be found in [16].

The independent task scheduling is usually defined as a four-objective optimization problem, with the makespan, flowtime, resource utilization and matching proximity as the scheduling criteria. In the multi-objective optimization two fundamental models are used, namely *hierarchical* and *simultaneous*. In the hierarchical case, the objectives are sorted according to their importance in the model and the optimization process starts from the most important criterion (which is usually makespan). In the simultaneous method all objectives are optimized simultaneously.

Recently in [7] and [8] the independent batch scheduling problem is addressed as a problem of optimal resource utilization from the Grid users' perspective under additional scheduling criteria: security and task abortion. The Grid scheduling problem is formalized as a non-zero sum game of the Grid users, who try to find the best assignment of their batch of tasks to resources. Two scenarios of the users' game are applied in this approach: (a) **non-cooperative symmetric game**, in which it is assumed that Grid users cannot cooperate with each other and (b) **a Stackelberg game**, which is an asymmetric two-level game, where there is one player (Leader), who has a privileged access to resources.

Each game has been translated into a hierarchical bi-level optimization problem. The joint cost of playing the game 'paid' by the users is defined as the objective for this problem. The cost of the particular user can be defined as the sum of four following componets[2]: (a) *Tasks execution cost*, which is calculated as an average completion time of the player's tasks on machines, to which they are allocated; (b) *Resource utilization cost*, which is calculated for each Grid user as an average idle time of machines on which his tasks are executed; (c) *Security cost*, which is defined as an average wasted time in the result of machine failures, because of the high security requirements and (d) *Task abortion cost*, which is expressed as an average wasted time in the result of tasks abortion on machines, because of Grid dynamics or special policies of the resource owners.

B. Resolution Methods

Heuristic methods are usually applied as the effective tools in Grid scheduling, which still remains challenging in the global optimization because of its complexity. The most commonly used heuristics are *ad hoc*, *local search-based*, *population-based* and their various hybrids.

Ah hoc Methods: These methods are usually used for single-objective optimization. They are simple and distinguished from their low computational cost, thus, they are also very useful in generating the initial solutions for population-based schedulers. The Ad-hoc heuristics could be grouped into an immediate mode heuristics and batch mode heuristics.

The *Immediate Mode Heuristics* group includes, among others, the following schedulers:

- *Opportunistic Load Balancing (OLB)*, where a task is assigned to the earliest idle machine without taking into account its execution time in the machine.
- *Minimum Completion Time (MCT)*, in which a task is assigned to the machine yielding the earliest completion time.
- *Minimum Execution Time (MET)*, in which a task is assigned to the machine having the smallest execution time for that task.

The *Batch Mode Heuristics* group contains, among others, the following methods:

- *Min-Min*: In this method for each task the machine yielding the earliest completion time is computed, then the task with the shortest completion time is selected and mapped to the corresponding machine.
- *Max-Min*: This method differs to the Min-Min in the final selection of the task with the latest completion time.

[2]Each component of the players' cost functions can be activated or not by the web application user. It means that the user can compose his own versions of the players' cost functions using the components needed to solve the problem specified in his scenario.

Table I
MAIN ATTRIBUTES OF GRID SCHEDULING

Attribute	Type	Brief description
Environment	Static	The number of resources is fixed and all of them are available
	Dynamic	The availability of the resources can dynamically change
Grid Architecture	Centralized	The schedulers have a full knowledge and control over resources
	Decentralized	No central entity controlling the resources, the local schedulers are responsible for managing and maintaining the tasks
	Hierarchical	The coordination of different schedulers at certain levels, the full knowledge of resources available for the schedulers at the lowest level
Task Processing Policy	Immediate	Tasks are scheduled as soon as they enter in the system
	Batch	Available tasks are sampled into a batch and the scheduler assign the batch to the resources
Tasks Interrelations	Independency	Tasks are scheduled independently of each other
	Dependency	There are precedence constraints among tasks

- *Sufferage*: The main idea of this method is to assign to a given machine a task, which would "suffer' more if it were assigned to any other machine.
- *Relative Cost*: In allocating tasks to machines, this method takes into account both the load balancing of machines and the execution times of tasks in machines.
- *Longest Job to Fastest Resource - Shortest Job to Fastest Resource (LJFR-SRFR)*: This method tries to simultaneously minimize both makespan and flowtime values: LJFR minimizes makespan and SJFR minimizes flowtime.

A comparative evaluation of the immediate, batch and ad hoc methods has been performed in [12].

Local search methods: These methods explore the optimization domain by starting from an initial solution and constructing a path in solution space. The most effective local-based Grid scheduler is Tabu Search (TS) due to its mechanisms of tabu lists, aspiration criteria, intensification and diversification (see [13]). TS can be easily hybridized with more sophisticated schedulers (like GAs) to improve their efficiency.

Population-based heuristics: In this methods a population of individuals, which is evaluated, crossed and mutated, is used to explore the solution space for the problem along a number of generations. The most popular in this group of methods are Genetic Algorithms (GA), addressed by many authors [11], [15]. Recently, a multi-population hierarchical GA-based scheduler has been proposed [6]. In this method a set of dependent genetic processes is executed simultaneously. Each process creates a branch in the tree structure of the whole strategy, by using the GA-based scheduler with different settings. The search accuracy in a given branch (expressed as the branch degree parameter) depends on the mutation probability set for the scheduler activated in this branch (the higher mutation prob.–the lower accuracy).

For solving game-theoretical scheduling a GA-based hybrid, namely GA-PMCT algorithm, has been defined [7]. In this method each component of the hybrid, i.e. GA and PMCT, operates as a scheduler at the global and players' levels, respectively. The main scheduling mechanism at the global level is defined as a GA-based scheduler. At the players' level the Players Minimum Completion Time-(PMCT) has been implemented, which is a modification of Minimum Completion Time (MCT). This component is responsible for the optimization of players' cost functions (separately for each player) [8].

III. INTEGRATION OF HYPERSIM-G GRID SIMULATOR WITH HEURISTIC-BASED SCHEDULERS

Using the simulators for the evaluation the Grid schedulers is feasible, mainly because of high complexity of the Grid environment. Many simulation packages, useful in the design and analysis of scheduling algorithms in Grid systems, have been recently proposed in the literature. Among many others, MicroGrid [10], ChicSim [9] and GridSim [2] seem to be the most popular in the domain. Some of them are integrated with the Grid portals in order to provide the users with an easy access to the simulation packages as well as the online monitoring of the scheduling process. A Web-based platform for simulating scheduling methods in Grid computing with Grid-Sim package was proposed in [5].

We used in our approach the HyperSim-G Grid simulator [14]. It extends the open source, discrete event simulation package HyperSim [4]. The main flow of HyperSim-G linked to the scheduling can be briefly described as follows. When a scheduling event is triggered, the simulator creates an instance of the scheduling problem, based on the current tasks and available machines pools. The instance contains: (a) workload vector of tasks; (b) computing capacity of machines; (c) prior load of machines and (d) the ETC matrix. The defined instance is then passed on to the selected scheduler which computes the planning of tasks to machines. Finally, the scheduler sends the planning back to the simulator, which makes the allocation and re-schedules any tasks assigned to machines not available in the system. In HyperSim-G the scheduling algorithms are completely

Table II
SETTING FOR THE GRID SIMULATOR FOR GENERATING STATIC INSTANCES IN SMALL GRID ENVIRONMENT

Init. number of hosts	32
Max. hosts	32
Min. hosts	32
Resource capacities (in MIPS)	$N(1000, 175)$
Add host	c(9999999999)
Delete host	c(9999999999)
Total number of tasks	512
Activation	Resource_and_time_interval(250000)
Scheduler strategy	$GA\_Scheduler(25, s)$
Reschedule	True
Workload of tasks	$N(250000000, 43750000)$
Interarrival	c(9999999999)
Host selection	All
Task selection	All
Number of runs	30

separated from the simulator, so the knowledge of the implementation of the specific scheduling methods is not needed.

Distributions: A number of distributions, including Constant distribution, Normal distribution, Uniform distribution, Exponential distribution, Zipf distribution, Triangle distribution and Trace distribution are implemented in the simulator. They are useful to model different scenarios in the Grid system, such as task arrival, machines joining or leaving the system, etc. In particular, by adequately choosing deviation values, we can model a wide range of Grid systems in terms of task heterogeneity, machine heterogeneity, consistency of computing, etc.

Use of Grid schedulers: The performance of each type of scheduler can be analyzed in two types of Grid environment: static and dynamic. In the static case, the number of tasks and the number of machines are kept constant during the simulation, while, in the dynamic case, the numbers of tasks and machines may vary over time. To exemplify this, in both cases we consider four typical Grid size scenarios for the experimental study: SMALL (32 hosts/512 tasks), MEDIUM (64 hosts/1024 tasks), LARGE (128 hosts/2048 tasks), and VERY LARGE (256 hosts/4096 tasks). The user can specify his own scenario changing the number of tasks and machines. The capacity of the resources and the workload of tasks are randomly generated by a normal distribution. It is also assumed that all tasks submitted to the system must be scheduled and all machines in the system can be used.

The simulator is highly parameterized in order to simulate the real-life Grid systems. An example of the simulator setting in the case of static scheduling in SMALL Grid environment is presented in Table II. In the setting, we have used constant distribution for the static case.

The number of hosts initially activated in the Grid environment is defined by the parameter *Init. number of hosts*. The parameters *Max.hosts* and *Min.hosts* specify the range of changes in the number of active hosts during the simulation process[3]. The frequency of appearing and disappearing resources is defined by given by *Add host* and *Delete host*, according to constant distributions for the static case, and normal distributions in dynamic case. The initial number of tasks is given by *Init. tasks*, which is kept constant in the static case. New tasks in the dynamic scheduling can arrive at the system with the frequency *Interarrival* until *Total tasks* is reached. The *Activation* parameter establishes the activation policy (it is usually modeled by an exponential distribution in the dynamic case). The assigned tasks which have not been executed yet cannot be rescheduled if the value of the boolean parameter *Reschedule* is false. The *Scheduler strategy* parameter denotes the Scheduler type. Its value $GA\_Scheduler(25, s)$ means that the simulator runs the GA-based scheduler for 25 seconds in simultaneous optimization mode[4].

For specified input parameters, the simulator outputs the values of makespan, flowtime, total potential time, expressed as the sum of available times of machines, total idle time of machines and total busy time of machines, resource utilization, etc. HyperSim-G can be run in many independent runs mode, in which the output results are averaged over the number of independent runs, and the standard deviation and confidence interval (95%) are computed. The detailed description of HyperSim-G structure and parametrization can be found in [14]. The simulation results and traces are graphically represented and stored at the server. They can retrieved in different file formats such as spreadsheet form or pdf files.

IV. WEB INTERFACE FOR HEURISTIC-BASED SCHEDULERS

The proposed Web application is based on the standard Client-Server architecture and LAMP (Linux + Apache + MySQL+ PHP) implementation. The general structure of the application is presented in Fig. 1.

The user plays role of the client of the application. He submits his requests by filling in the formulaires in order to specify the scheduling attributes, the general parameters of the simulator and the parameters of selected scheduler. The user's requests are processed by the "Management" module and then managed as a queue by the Condor system [3] in order to allow the multiple simultaneous executions of the application and simulator by many users and to monitor the state of the submitted simulations. The application administrator is able to manage users, monitor executions, set up some global parameters (maximum number of executions, maximum number of hosts, tasks) etc.

To run the simulator each user must firstly register in the system by selecting the 'Register' option on the main

[3]In the case of dynamic scheduling, they are different from the initial number of hosts.

[4]Similarly, the parameter h can be used to indicate hierarchic mode optimization, e.g. $GA\_Scheduler(25, h)$.

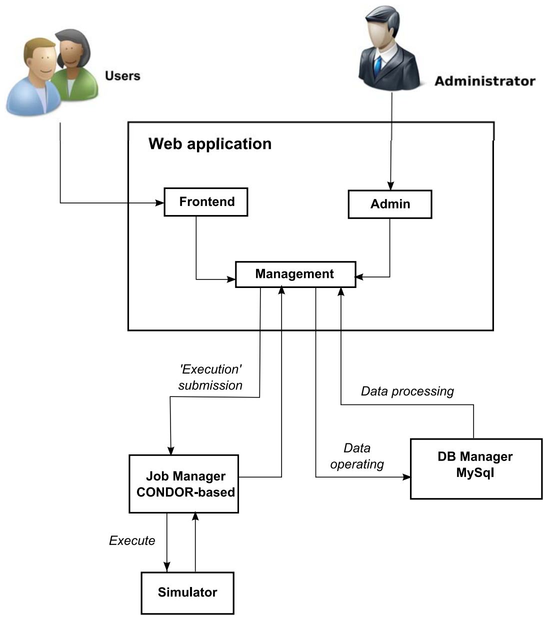


Figure 1. The architecture of Web application.

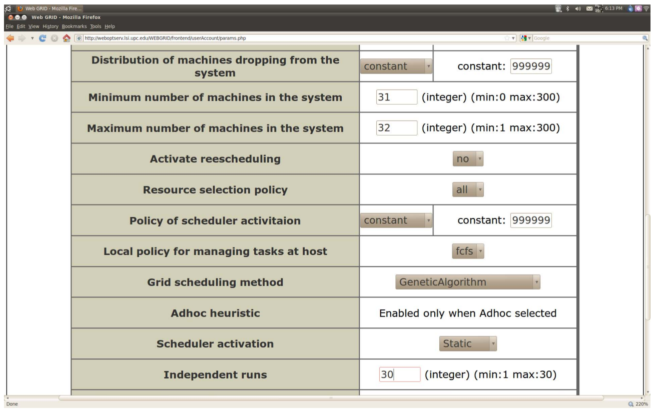
page of the application. Registering is also necessary for checking and analysis (by the particular user as well as by the administrator) of the current and previous simulation results. The results are graphically represented, they can be exported as a spreadsheet or pdf document. All users' account details and the results of his simulation are stored in the application database.

A. *Experimental study example: Evaluation of the GA-based scheduler in the SMALL Grid scenario*

In this section we show by example the use of the Web application in the evaluation of GA-based schedulers in the SMALL Grid scenario. Once logged into the application (after selecting the 'Login Simulator' option on the main page), through the option of New execution the user can set the simulator parameters specified in Table II by filling in the formulaire 'Simulator parameters'. The snapshot of this formulaire is presented in Fig. 2.

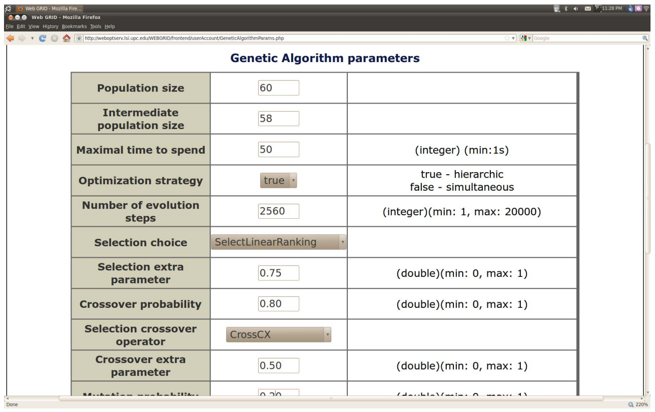
The user can the select the type of the scheduler (e.g. GA scheduler) and confirm parameters setting by pressing the 'Next' button. It leads the user to the 'Genetic Algorithm parameters' formulaire, the snapshot of which is presented in Fig. 3. The algorithm parameters values in our experiment are specified in Table III.

The simulation is activated by pressing the 'Run' button on the last form. By selecting View result option, the user can consult the simulation result as well as the corresponding trace of the simulation (see Fig. 4). The table with experimental results on the Web page reports the averaged, maximal and minimal values of the main schedulers performance metrics achieved in the 30 independent runs of the simulator


Web GRID - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://weboptserv.lsi.upc.edu/WEBGRID/frontend/userAccount/params.php

Distribution of machines dropping from the system	constant	constant: 999999
Minimum number of machines in the system	31 (integer) (min:0 max:300)	
Maximum number of machines in the system	32 (integer) (min:1 max:300)	
Activate reescheduling	no	
Resource selection policy	all	
Policy of scheduler activitaion	constant	constant: 999999
Local policy for managing tasks at host	fcfs	
Grid scheduling method	GeneticAlgorithm	
Adhoc heuristic	Enabled only when Adhoc selected	
Scheduler activation	Static	
Independent runs	30 (integer) (min:1 max:30)	

Figure 2. The snapshot of the simulator parameters formulaire.


Web GRID - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://weboptserv.lsi.upc.edu/WEBGRID/frontend/userAccount/GeneticAlgorithmParams.php

Genetic Algorithm parameters

Population size	60	
Intermediate population size	58	
Maximal time to spend	50	(integer) (min:1s)
Optimization strategy	true	true - hierarchic false - simultaneous
Number of evolution steps	2560	(integer)(min: 1, max: 20000)
Selection choice	SelectLinearRanking	
Selection extra parameter	0.75	(double)(min: 0, max: 1)
Crossover probability	0.80	(double)(min: 0, max: 1)
Selection crossover operator	CrossCX	
Crossover extra parameter	0.50	(double)(min: 0, max: 1)

Figure 3. The snapshot of the Genetic Algorithm parameters formulaire.

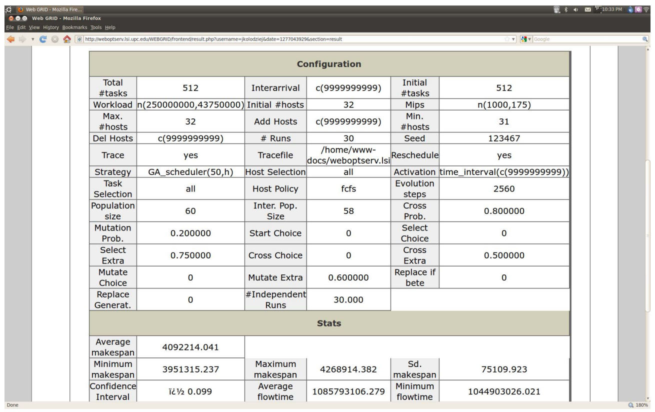
under the same configuration of scheduler's and simulator's parameters. The results of the simulations as well as their traces can be graphically represented (see Fig. 5).

V. Conclusions and Future Work

In this work we have presented a Web interface for the HyperSim-G Grid simulator, which allows the user to select, through a user-friendly interface different scheduler types. The scheduler types include ad hoc, local search

Table III
GA settings for static scheduling in SMALL Grid scenario

Parameter	
evolution steps	2560
population size	60
intermediate pop.	58
selection method	LinearRanking
crossover method	CX
cross probab.	0.8
mutation method	Rebalancing
mutation probab.	0.2
$replace\_only\_if\_better$	false
$replace\_generational$	false
initialization	LJFR-SJFR



Configuration					
Total #tasks	512	Interarrival	c(9999999999)	Initial #tasks	512
Workload	n(250000000,43750000)	Initial #hosts	32	Mips	n(1000,175)
Max. #hosts	32	Add Hosts	c(9999999999)	Min. #hosts	31
Del Hosts	c(9999999999)	# Runs	30	Seed	123467
Trace	yes	Tracefile	/home/www-docs/weboptserv.lsi	Reschedule	yes
Strategy	GA_scheduler(50,h)	Host Selection	all	Activation	time_interval(c(9999999999))
Task Selection	all	Host Policy	fcfs	Evolution steps	2560
Population size	60	Inter. Pop. Size	58	Cross Prob.	0.800000
Mutation Prob.	0.200000	Start Choice	0	Select Choice	0
Select Extra	0.750000	Cross Choice	0	Cross Extra	0.500000
Mutate Choice	0	Mutate Extra	0.600000	Replace if bete	0
Replace Generat.	0	#Independent Runs	30.000		
Stats					
Average makespan	4092214.041				
Minimum makespan	3951315.237	Maximum makespan	4268914.382	Sd. makespan	75109.923
Confidence Interval	ï¿½ 0.099	Average flowtime	1085793106.279	Minimum flowtime	1044903026.021

Figure 4. The snapshot of the simulation results.

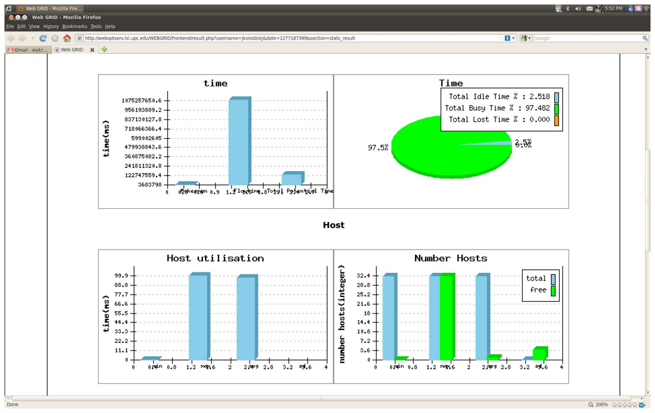


Figure 5. The snapshot of the graphical representation of the simulation results.

and population based schedulers. The Web application is meant to facilitate the experimental study and analysis of the performance of Grid scheduling using various meta-heuristics as schedulers. Additionally, the users can trace the state of the simulations, export the simulation results and make graphical representation of the results, which can as well be received by email. The application is based on Condor queueing system, which enable the multiple simultaneous runs of the simulator by many users with fast response time. We have exemplified the usefulness of the approach for the experimental evaluation of Grid schedulers through a step-by-step procedure for the case of the GA-based scheduler in several Grid scenarios.

References

[1] S. Ali, H.J. Siegel, M. Maheswaran and D. Hensgen: "Task execution time modeling for heterogeneous computing systems", *Proceedings of Heterogeneous Computing Workshop*, 2000, pp. 185–199.

[2] R. Buyya and M.M. Murshed: "Grid-Sim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing", *Concurrency and Computation: Practice and Experience*, 14(13-15), 2002, pp. 1175-1220.

[3] Condor High Throughput Computing System: http://www.cs.wisc.edu/condor/

[4] http://opensimulator.org/wiki/Hypergrid

[5] O. Kang and S. Kang: "Web-based Dynamic Scheduling Platform for Grid Computing", *International Journal of Computer Science and Network Security*, 6(5), 2006, pp. 67-75.

[6] J. Kołodziej, F. Xhafa and Ł. Kolanko: "Hierarchic Genetic Scheduler of Independent Jobs in Computational Grid Environment", *Proc. of 23rd ECMS, Madrid, 9-12.06.2009, in J. Otamendi, A. Bargieła, J.L. Montes and L.M. Doncel Pedrera eds.*, IEEE Press, Dudweiler, Germany, 2009, pp. 108–115.

[7] J. Kołodziej and F. Xhafa: "A Game-Theoretic and Hybrid Genetic meta-heuristic Model for Security-Assured Scheduling of Independent Jobs in Computational Grids", *Proc. of CISIS 2010, Cracow, 15-18.02.2010, in L. Barolli, F. Xhafa and S. Venticinque eds.*, IEEE Press, , USA, 2010, pp. 93–100.

[8] J. Kołodziej, F. Xhafa and M. Bogdański: "Secure and Task Abortion Aware GA-based Hybrid Metaheuristics for Grid Scheduling", Accepted for publication in *Proc. of PPSN XI, September 10-15, 2010, Cracow, Poland*, LNCS, Springer Vlg.

[9] K. Ranganathan and I. Foster: "Simulation studies of computation and data scheduling algorithms for data grids", *J. Grid Comput.*, 1(1), 2003, pp. 53-62.

[10] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien: "The micr量grid: a scientific tool for modeling computational grids", *Journal of Sci. Program.*, 8(3), 2000, IOS Press, pp. 127-141.

[11] F. Xhafa and A. Abraham: "Meta-heuristics for Grid Scheduling Problems". In *Meta-heuristics for Scheduling in Distributed Computing Environments*, Chapter 1, Series *Studies in Computational Intelligence*, Springer Vlg., 2009, pp. 1–37.

[12] F. Xhafa, L. Barolli and A. Durresi: "Batch Mode Schedulers for Grid Systems". *International Journal of Web and Grid Services*, 3(1), 2007, 19–37,.

[13] F. Xhafa, J. Carretero, E. Alba, B. Dorronsoro: "Tabu Search Algorithm for Scheduling Independent Jobs in Computational Grids", *Computer And Informatics Journal, special issue on "Intelligent Computational Methods"*, J.Burguillo-Rial, J.Kołodziej and L. Nolle eds., 28(2), 2009, pp 237–249.

[14] F. Xhafa, J. Carretero, L. Barolli and A. Durresi: "Requirements for an Event-Based Simulation Package for Grid Systems". *J. of Interconnection Networks*, World Sci. Pub., 8(2), 2007, pp. 163–178.

[15] F. Xhafa, J. Carretero, A. Abraham: "Genetic Algorithm Based Schedulers for Grid Computing Systems". *International J. of Innovative Computing, Information and Control*, (5), 2007, pp. 1–19.

[16] Xhafa, F., Abraham, A.: "Computational models and heuristic methods for Grid scheduling problems", *Future Generation Computer Systems*, 26, 2010, pp. 608–621.