CMST 13(2) 121-129 (2007) DOI:10.12921/cmst.2007.13.02.121-129

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **13(2)**, 121-129 (2007)

GSSIM – Grid Scheduling Simulator

Krzysztof Kurowski [1], Jarek Nabrzyski [1]
Ariel Oleksiak [1], Jan Weglarz [2]

[1]*Poznan Supercomputing and Networking Center*
e-mail: {krzysztof.kurowsk/naber/ariel}@man.poznan.pl
[2]*Institute of Computing and Management Sciences*
Poznan University of Technology
jan.weglarz@cs.put.poznan.pl

(Rec: November 26, 2007)

Abstract: Grid simulation tools provide frameworks for simulating application scheduling in various Grid infrastructures. However, while experimenting with many existing tools, we have encountered two main shortcomings: (i) there are no tools for generating workloads, resources and events; (ii) it is difficult and time consuming to model different Grid levels, i.e. resource brokers, and local level scheduling systems. In this paper we present the Grid Scheduling Simulator (GSSIM), a framework that addresses these shortcomings and provides an easy-to-use Grid scheduling framework for enabling simulations of a wide range of scheduling algorithms in multi-level, heterogeneous Grid infrastructures. In order to foster more collaboration in the community at large, GSSIM is complemented with a portal (http://www.gssim.org) that provides a repository of Grid scheduling algorithms, synthetic workloads and benchmarks for use with GSSIM.

Key words: Grid scheduling, resource management

I. INTRODUCTION

Grid scheduling algorithms have been a subject of intense research over the last decade [13] and [14]. However, evaluation and comparative analysis of these algorithms and research experiments are often difficult to perform. This is caused by many problems, including, for example, difficulties in obtaining exclusive access to large scale infrastructures for research purposes or lack of certain functionalities of real resource management systems, such as advance reservation (AR) or Grid user accounting. Therefore, Grid scheduling algorithms have been often tested in simulation environments.

Nevertheless, in order to perform a reliable simulation experiment researchers must cope with several issues. Workloads usually come from single and often independent local clusters, collected under specific conditions, and do not contain information about workflows, co-allocation requests etc. Additionally, available simulation environments usually do not allow simulating multiple autonomous scheduling elements, such as Grid and local schedulers. As a consequence of all these problems researchers have a limited chance to reuse and compare results of their analysis.

Therefore, we expect from a simulation environment to provide flexible and easy way to describe, generate and share input data to experiments. The generator should enable generation of workload and resources including different probability distributions in specific time periods, correlations between attributes, etc. To ensure interoperability a simulation environment must support standard workload formats and enable relatively easy replacement of job and resource descriptions. Furthermore, to make the simulation environment more similar to realistic behavior resource failure generation should be included.

The second group of requirements is related to an architecture and a generic model that should enable building multilevel environments and using various scheduling strategies with diverse granularity and scope. In particular, researchers should be able to build architectures consisting of two tiers in order to insert scheduling algorithms both to local schedulers and grid brokers. Additionally, simulator should provide means to schedule various types of applications: from processes of single tasks (e.g. MPI applications) up to the whole workflows. It is also important to support features such as simulation of network, performance estimations (with possible predefined errors), and custom calculation of execution time.

In this paper, we attempt to address these issues and propose Grid Scheduling SIMulator (GSSIM) built on top of GridSim [1]. GSSIM has been designed as a simulation framework which enables easy-to-use experimental studies of various scheduling algorithms and meet requirements listed above. The workloads used are compliant with known workload formats such as Standard Workload Format (SWF) [2] and Grid Workload Format (GWF) [3].

To enable sharing of the workloads, algorithms and results, we propose a GSSIM portal [15] where researchers may download various synthetic workloads, resource descriptions, scheduling plugins, and results [15]. This portal complements other known websites related to this area since it provides a repository of synthetic workloads, including the online services for generation of workloads, and scheduling plugins.

The remaining part of the paper is organized as follows. In Section II the related work is presented. Section III presents an overall GSSIM architecture. In Section IV we explained how workloads and resources are modeled. Section V contains a description of the interfaces needed to implement scheduling algorithms.In Section VI an example of GSSIM use for the simulation of a simple scheduling problem is illustrated. Section VII concludes this paper.

II. RELATED WORK

Due to the complexity and costs of building and operating Grid testbeds, extensive research has been conducted in the area of computer-based simulation tools. A comprehensive taxonomy for design of simulation tools to model large and distributed systems has been presented in [4]. One of many categories, called *Grid Scheduling Systems*, was identified. Four simulation tools: Bricks, MicroGrid, SimGrid and GridSim were classified into this category. Bricks [5] simulates various behaviors of Grid computing systems such as the behavior of networks and resource scheduling algorithms. However, Grid environments modeled by Bricks are based on a relatively simple client-server architecture typical, for instance, of distributed systems providing a remote access to scientific libraries and packages running on high-performance computers. The second tool, called MicroGrid [6] was classified as an emulator and (not a simulator). SimGrid [7] aims at providing the right model and level of abstraction for studying Grid-based scheduling algorithms and generates correct and accurate results but it offers fewer simulation capabilities compared to GridSim. Looking at a classification given in [4] it does not provide system support such as in GridSim, its simulation engine is serial, and is not object-oriented as it is implemented in C. GridSim adopts the multi-layered design architecture and is based on event simulation software called SimJava [8]. It extends various SimJava packages to offer a high degree of modeling and simulation of heterogeneous Grid resources (both time- and space- shared), users, applications, brokers and schedulers in a Grid computing environment. It is worthy of note that GridSim also supports simulation of Advance Reservation (AR) mechanisms [9]. Moreover, the latest version contains comprehensive support for simulating data Grids and market models, such as auctions. However, it contains neither language-based design environment nor workload generation system support [4]. Although many simulation tools have been built, none of them satisfied all our requirements especially concerning experimental data generation and flexible development and sharing of various both Grid- and local-level scheduling plugins. This fact encouraged us to develop GSSIM using some existing software components.

In parallel to the development of simulation and emulation tools, many projects have tried to tackle problems around modeling synthetic and real workloads and finally simulating them under various Grid environment assumptions. Probably the first detailed model of parallel workloads was proposed by Feitelson in 1996 [10]. Since then, a lot of workload data have been collected [2], analyzed and modeled [11]. Real traces on production Grids are being collected [3] and have not yet become widely available. Models for generating synthetic Grid workloads, on the other hand, are still emerging and are subject to further research. Therefore, we have designed GSSIM in a way that enables researchers not only to create and test various Grid and local level scheduling algorithms but also to generate synthetic and adopt real workloads.

III. GSSIM ARCHITECTURE AND MODEL

The GSSIM framework is based on GridSim and SimJava2 packages. However, it provides a layer added on top of the GridSim adding capabilities to enable easy and flexible modeling of Grid scheduling components. GSSIM also provides an advanced generator module using real and synthetic workloads. The overall architecture of GSSIM is illustrated in Fig. 1.

GSSIM distinguishes between two types of scheduling components: Grid brokers and resource providers. As shown in Fig. 1 multiple scheduling strategies may be plugged into both levels. Input data can be read from real sources or generated using the generator module. The major extensions of GridSim, namely input data modeling

and scheduling interfaces, are described in the subsequent sections.

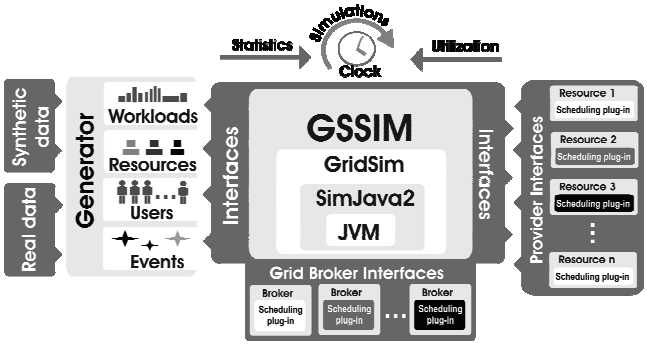

Fig. 1. Generic GSSIM architecture

One of the major GSSIM objectives is flexibility in terms of using a variety of Grid scheduling strategies and workloads. However, Grid jobs may have various shapes and levels of complexity ranging from workflows, through large-scale parallel applications, up to single tasks that require single resources. Depending on type of Grid jobs, scheduling strategies may have different scope and need different input data. Therefore, to make development of scheduling plugins easier on one hand and keep it flexible on the other, we distinguish in GSSIM several levels of information about incoming jobs. These levels are presented in Fig. 2. We assumed that there is a queue of jobs submitted to a Grid scheduler. Each job consists of one or more tasks. Thus if preceding constraints are defined a job may be a whole workflow. More details about generation of this model and its application to implementation of scheduling plugins are presented in Sections IV and V, respectively.

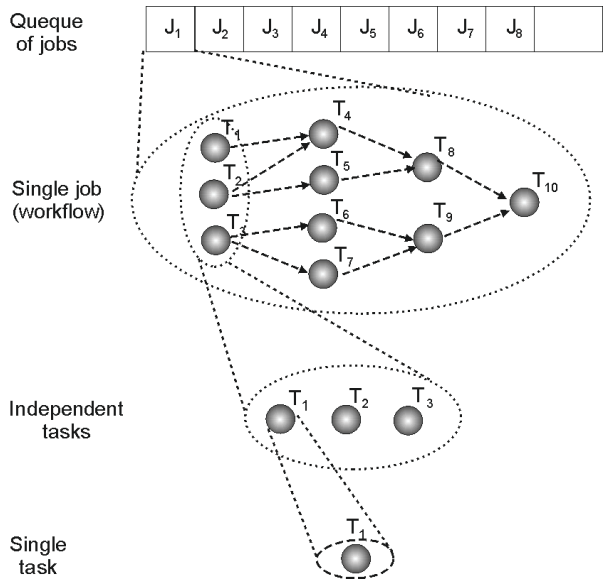

Fig. 2. Levels of information about jobs

IV. INPUT DATA MODELING

In general, input data in GSSIM consist of workload and resource descriptions. Users may both generate new or read the existing synthetic data. Third party real workloads can also be imported by GSSIM. If any parameters are missing after importing a workload, they can be generated by GSSIM and added.

If synthetic workload and resource description are used, for each generated parameter the following probabilistic attributes and constraints can be specified: *avg* – mean *value*, *stdev* – standard deviation, *min*, *max* – minimal and maximal value, *seed* – seed for random process, *distribution* – probabilistic distribution, and *startTime*, *endTime* – time period.

The definition of a seed allows to obtain the same random values in different experiments. The following distributions are currently implemented in GSSIM: *constant*, *normal*, *poisson, exponential* and *uniform*.

Description and generation of workload and resources are presented in more detail in the following sections.

IV.1 Workload

Workload contains information about jobs, their structure, resource requirements, relationships, time intervals etc. We assumed a model in which each job consists of one or more tasks. A job may contain preceding constraints between tasks (workflow) as it was said in Section III. The description of how this job model adopted is handled by the scheduling interface is presented in Section V.1. The next sections provide information on how workloads are described and generated in GSSIM.

IV.1.1. Workload Description

Generally, we can distinguish between two parts of the workload description: a basic workload description using Grid Workload Format (GWF) and extensions described in XML files. The former is mostly needed by a core simulator to perform simulations while the latter are input data to scheduling algorithms (plugins). For instance, a simulator must have some knowledge about task runtimes to perform simulation correctly, while many scheduling algorithms assume lack of this knowledge. On the other hand, some parameters used by scheduling strategies such as preferences of users are not needed by a core simulator.

XML-based job descriptions are passed to scheduling plugins. In order to describe job descriptions we adopted formats used in GRMS [13]. We made this decision for two main reasons. First, this schema is comprehensive enough and it offers various extensions for workflows, time constraints, user preferences and other useful information. Secondly, this choice allows us to easily move algorithms between our real Grid environments managed by GRMS and simulation environment (GSSIM) minimizing amount of code modifications. Thus, we can easily use the same algorithms in practice, after modeling them on the simulator.

If researcher needs a synthetic workload he can use the flexible GSSIM workload generation capability, presented below.

IV.1.2. Workload Generation

Parameters that characterize workload generation are specified in the XML-based configuration file. This file allows to define basic parameters such as job count, arrival rate, task count, task runtimes, and simulation time. Requirements concerning all these parameters can be specified using attributes defined at the beginning of Section 4. Definition of workload parameters using independent distributions may result in non-realistic workloads named in [12] as "naive". For instance, in [11], the authors argue that a *Poisson* distribution does not sufficiently model a jobs arrival process since it does not take into consideration daily cycles. To compensate for this deficiency, the GSSIM workload generation file enables defining different distributions in specific daytime periods. For example, one can define two distributions: one for thr day and one for the night. Another parameter whose modeling is not trivial is runtime. A correlation of runtime with task size (number of requested processors) was discovered in several studies. Therefore, GSSIM provides attributes that allow defining a correlation between runtime and size. A simple linear dependency is used as shown in the following formula:

$$runtime(s_j) = \alpha s_j + \beta$$

where s_j is a size of a task j, α and β are parameters given by a workload designer. Both parameters can be defined by probabilistic attributes defined above.

All these mechanisms allow users to generate synthetic workloads. Of course, more complex methods are often needed to model workloads accurately and realistically. GSSIM tools provide an easy way to generate input data for at least initial and high level evaluation of algorithms. Moreover, these generated workloads may provide a basis for further tuning in order to improve their quality. Finally, the experiments conducted in [12] shown that a choice of workload alone (e.g. real against synthetic one) does not cause significant changes in the evaluation of scheduling algorithms. Based on these observations we believe that the GSSIM workload generation feature is an easy but also useful and reliable way of providing input data for scheduling experiments. Additionally, runtime calculation plugins may be implemented to model runtimes in a more complex way. These plugins return task runtime based on parameters of available resources and a given task. This capability allows taking into account the performance models of applications and influence of heterogenous resources on runtime.

In addition to basic parameters of a workload, a GSSIM user may also specify details of generation of other, more complex elements. They include task preceding constraints (in order to define workflows), hard constraints (to specify resource requirements), soft constraints (to express users' preferences), and time constraints (to define scheduling requirements).

IV.2. Resources

The second part of the input data that must be delivered for simulations is the description of resources. In GSSIM, it generally consists of two types of information: definitions of resource providers (autonomous systems that represent separate administrative domains) and network topology. The former contains a structure and parameters of resources, and information whether they support advance reservation mechanisms. The latter basically defines network links and their parameters. GSSIM also allows defining the required characteristics of resource failures. Similarly to a workload, resource descriptions may be both provided by a developer or generated. In the latter case, a resource description configuration file is used.

IV.2.1 Resource Description

Resources in GSSIM are also described using an XML-based format. The description contains definitions of all resource providers available in an experiment. Each resource provider consists of three main elements. First element is a list of queues including their parameters, e.g. priority, number of processors assigned to a queue, etc. The second element includes collective information about resources available at a given resource provider or description of all single machines. Each description contains information about resource parameters, e.g. number of free CPUs, memory, etc. Finally, availability of AR mechanism is indicated in the description.

Since GSSIM supports two-level scheduling architectures, in which local schedulers may have their own scheduling strategies, the resource description also contains information about assignment of specific algorithms (plugins) to particular resource providers.

As said above, GSSIM enables automatic generation of resource descriptions. The generation process is similar to that described in Section IV.1. The configuration file contains a definition of parameters related to: the total number of processors, amount of memory, number of queues and their basic parameters such as priority, maximum number of jobs etc. In addition, parameters concerning network connections such as bandwidth and delay may be generated. For each resource element, failure frequency can be specified (see Section II.3 for more details).

IV.2.2. Generation of Failures

In dynamic and cross-domain infrastructures such as Grids resource failures are common events. Therefore, Grid middleware should be able to correctly handle them. To this end, tests of Grid scheduling algorithms should take into account high probability of errors.

GSSIM allows researchers to specify probability of failure for every single resource. In more detail, for each resource element in the configuration file one can specify *failure frequency* τ, which is defined as the expected number of failures per time unit, and *unavailability time* μ. The latter parameter allows to define the time during which a resource is unavailable, e.g. time that resource needs to recover from a failure. For instance, if τ = 0.000001 and μ = 100, a given resource will fail once every 1 000 000 time units on average and its each period of unavailability will take 100 time units on average.

Based on the parameters in the configuration file a list of resources' unavailability is generated. Of course, the generated list may be modified or even replaced by a list given by a researcher, e.g. prepared on the basis of real observations.

When a resource fails it is removed from a list of resources for a defined period of time and an appropriate event is sent to the given local provider plugin (see Section V). This functionality enables testing a reliability of algorithms and should lead to development of more robust and self-healing scheduling strategies.

V. SCHEDULING INTERFACE

This section contains description of scheduling interfaces at Grid and local levels. Each scheduling plugin must implement one of them.

V.1. Grid Scheduler

This interface simulates an environment of a Grid scheduler. It provides all necessary information needed to schedule jobs in Grids and imposes implementation of basic functionality required from Grid schedulers.

In general, the major methods of the interface responsible for handling different types of events is *schedule*, which performs scheduling when specific event occurs.

This method enables implementing various scheduling strategies: off-line scheduling for whole sets of incoming jobs, dynamic scheduling based on specific events, periodic rescheduling, etc. The following events relevant for Grid scheduler have been considered in GSSIM: TIMER, JOB_ARRIVED, TASK_FINISHED, TASK_FAILED, TASK_CANCELED, RESOURCE_FAILED, PROVIDER_FAILED, RESERVATION_ACTIVE, RESERVATION_FINISHED, RESERVATION_FAILED. Of particular importance are events TIMER and JOB_ARRIVED that enable scheduling periodically and whenever new jobs arrive, respectively. This interface method returns scheduling decisions that contain information about selected resources and, in case of scheduling based on advance reservation, identifiers of reservations.

Additionally, to make the development of plugins easier and more focused, the following methods are available according to the levels defined in Fig. 2: *scheduleJobs*, *scheduleJob*, *scheduleTasks*, and *scheduleTask*. The advantage of this method is that the authors of scheduling plugins may choose at which level they implement their algorithm (i.e. which method to override). For instance, if a scheduling algorithm is focused on matching single tasks to resources, then only *scheduleTask* need to be implemented. If an algorithm schedules all jobs at once only the *scheduleJobs* method must be overwritten.

In addition to the information about incoming jobs, a Grid scheduler needs knowledge about the environment. In GSSIM, Grid scheduling plugins have access to the information about running jobs, resources, network topology, reservations, and estimations of runtime and resource requirements. A reservation entity provides information about reservations and allows a Grid scheduling plugin to request and negotiate specific reservations. Particular strategies may use various levels of knowledge about resources. Basic scheduling strategies base their decisions on very limited information while more advanced algorithms can apply knowledge about running jobs, performance predictions, and network topology to schedule jobs efficiently. Grid scheduler may perform best-effort scheduling based on available information about resources or apply scheduling with QoS by negotiating offers from resource providers.

V.2. Local Schedulers

At the level of local schedulers there are two types of interfaces distinguished in GSSIM. They correspond to two different types of scheduling approaches in local systems. The "basic" interface is based on a "best effort" approach where tasks are submitted to local queues and no guarante is given concerning the start time, resource availability, etc. The second type assumes scheduling with QoS guarantees. In this case, resource providers advertise and possibly negotiate their offers and, if they are successful, reserve requested resources for a certain period in the future using advance reservation mechanisms. These both interfaces are presented in the subsections below.

V.2.1. Basic Scheduling Interface

This interface provides queue management mechanisms for plugin developers. It consists of the following methods: *scheduleNewTasks*, *scheduleOnEvent*, and *scheduleCyclic*. *scheduleNewTasks* schedules the newly arrived tasks, i.e. executes them or puts into a queue. *scheduleCyclicis* invoked periodically. In the *scheduleOnEvent* method a type of an event is passed according to the list defined in Section IV. Input data for scheduling methods of this interface include a list of tasks being executed, queues, and state of resources.

V.2.2. Interface for QoS-based Scheduling

This interface provides advance reservation and negotiation mechanisms for plugin developers. It distin guishes initial and committed reservations. In this way it enables development of reservation protocols based on two-phase commit. Major methods of the interface are presented in Table 1.

Table 1. Methods of the interface with QoS support

Methods	Descriptions
getOffers	Returns reservation offers based on time and resource requirements
checkOffers	Returns reservation offers based on counteroffers from resource consumer
initReservation	Initially reserves a requested slot or reject the request; if the reservation is not committed before certain time the initial reservation expires
commitReservation	Commits reservation or rejects the request
getStatus	Returns status of reservation
cancelReservation	Cancels a reservation
modifyReservation	Decides whether the reservation can be modified
submitTasks	Executes tasks on reserved resources

Input data for scheduling methods of this interface include a list of reservation requests, lists of existing committed and initial reservations, tasks being executed, queues, and state of resources. Additionally, time and resource requirements, and proposed offers are passed to methods responsible for negotiations of reservations.

Both interfaces described above are complementary to each other which means that they can be used together in a single resource provider. Their use and implementation depends on specific scheduling strategies.

V.3. Scheduling Problems handled by GSSIM

Using the interfaces and workloads presented above GSSIM enables simulations of diverse scheduling strategies applied to various types of applications. With regard to classes of Grid level applications it is possible to schedule according to the generic model presented in Fig. 2. That means that GSSIM can simulate scheduling of multiple independent tasks at once, parallel tasks, and whole workflows. GSSIM also enable simulating of various scheduling problems. In particular, both best-effort and QoS-based approaches are available. To realize the latter case, GSSIM supports negotiations between Grid schedulers and resource providers, advance reservation mechanism with two phase commit usage. Additionally, GSSIM provides possibility of scheduling based on performance estimations. These estimations can be generated on the basis of runtimes included in the workload or using custom algorithm implemented by researcher. At a local level, a developer of a scheduling plugin has unlimited access to queues and running tasks. Therefore, a variety of both space- and time- sharing policies may be applied. For instance, developers can implement algorithms based on backfilling or preemption using arbitrary types of resources can be scheduled (not only processors). Both shared and distributed memory cases are supported.

For each experiment detailed results are collected. They contain many basic criteria commonly used in evaluation of scheduling algorithms, e.g. makespan, mean task completion time, etc. One of possible scheduling experiments together with obtained results is illustrated by a short example given in the next section.

VI. EXAMPLE OF EXPERIMENT

In this section we present an example of the simple experiment conducted using the GSSIM framework. Its

goal is to illustrate steps and elements needed to perform simulations in GSSIM. The experiment is a test of two basic algorithms on a Grid and local level: *MPL* and *FCFS*, respectively. The former, *Min–Parallel–Load* (*MPL*), selects a resource provider with the lowest parallel load per processor (the sum of job sizes over number of available processors), i.e.

$$MPL = \min \frac{\sum_{j=0}^{j=r_k} S_j}{p_k}$$

where r_k is a number of tasks at resource provider k, p_k denotes a number of available processors, and s_j is a size of task j, or a number of requested processors. The latter, First Come First Served, allocates tasks to resources of a local system in the order of their arrival. Both strategies are algorithms commonly used for comparison purposes.

Resources are delivered by two resource providers. Each of them is controlled by a queueing system with a single queue available. The generated workload consists of 10 independent jobs.

This section contains examples of the following elements of the experiment: workload description, resource description, local scheduler plugin, Grid scheduler plugin, and results.

VI.1. Workload Description

GSSIM generates a workload that consists of 10 jobs. Each task requires from 1 up to 4 processors where the values are generated using a uniform distribution. To generate a task length a normal distribution was used. The task length denotes a number of operations needed to perform to complete a task. It can be also interpreted as a task runtime at a machine that performs a single operation per one time unit. The arrival rate has a Poisson distribution with an average equal to 50 time units.

VI.2. Resource Description

For each resource provider a queueing system containing a single queue is described. The first resource provider provides access to a total number of 4 processors while the second one to 8 processors. All the processors provided by both providers have the same speed. None of the defined resource providers supports advance reservation.

VI.3. Local Scheduler Plugin

In this section a simple example of a local scheduler plugin is presented. It implements FCFS algorithm, so if a new task comes, the plugin algorithm either puts it at the end of a queue or, if there are free resources that meet resource requirements, executes the task. If any task has finished, plugin tries to execute subsequent tasks from a queue. To this end two methods must be implemented: *scheduleNewTasks* and *scheduleOnEvent*. A simplified version of the former is presented in Fig. 3.

```
public int scheduleNewTasks (SubmittedTask[] newTasks,
                             List<SubmittedTask> inExecution,
                             Queue<SubmittedTask>[] queues,
                             Resource resource) {
    Queue<SubmittedTask> queue = queues [0];
    SubmittedTask task = newTasks[0];
    Map freeRes = resource.getFreeResources(
                  task.getResourceRequirements());
    if (freeRes != null)
        inExecution.add(task);
    else
        queue.add(task);
    return 1;
```

Fig. 3. The *scheduleNewTasks* method of the local scheduler plugin

VI.4. Grid Scheduler Plugin

The last element of the experiment is a Grid scheduler plugin. It corresponds to an algorithm of Grid scheduler. It applies simple *MPL* strategy, which assigns every task to a resource provider that has the lowest sum of task sizes over number of available processors.

VI.5. Results

In this example a set of 10 tasks has been generated. Basic characteristics are presented in the table in Fig. 4.

Applying the scheduling strategies analytically to this set of tasks led to the schedules illustrated below the table in Fig. 4.

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Size	2	4	1	3	1	2	4	3	1	1
Length	1079	1401	1396	2127	1322	1195	2329	2063	818	1078
Arrival	0	36	134	191	253	272	316	394	447	439

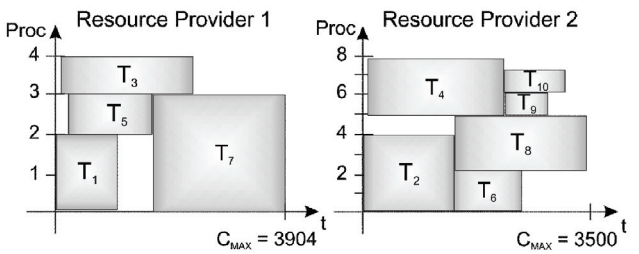

Fig. 4. Schedules of tasks in the example

Major statistics collected during this GSSIM experiment are as follows: *Job start time* = 969.9, *Job completion*

time = 2504.7, *Job execution time* = 1534.8, *Job waiting time* = 923, *Makespan* = 3904, *Resources total load* = 0.92, and *Resources queue length* = 0.48. In addition to mean values presented above, for each of parameters multiple statistical measures are calculated: standard deviation, variance, minimum, maximum, sum, and count. Additionally, GSSIM enables researchers to repeat experiment multiple times to be sure that obtained results are statistically representative. In this case, GSSIM results contains statistics collected during multiple runs of experiment.

More details about this and other experiments, which are useful for validation of GSSIM results and as examples for other users, are provided within the Grid Scheduling Simulations portal.

VII. CONCLUSION

In this paper we present a tool, which is an attempt towards more common and easier comparison of results of Grid scheduling experiments. It is complementary to the existing activities with regard to two major issues. First, it allows researchers to share all elements of experiments including implementations of Grid scheduling algorithms (not only workloads). Second, in order to compensate for a low number of currently available real Grid workloads, it provides generation tools and a repository for synthetic and semi-synthetic workloads. Furthermore, since GSSIM supports both SWF and GWF formats, it is compatible with other existing approaches. However, due to requirements for a high flexibility of the simulator, it also provides extensions allowing to describe additional elements of a workload.

Among many possible future works, we are eager to adopt other standard formats provided especially if their set is sufficiently flexible and comprehensive (e.g. job and resource description languages from OGF). Further, we plan to validate GSSIM by comparing results obtained in GSSIM experiments with those performed in real environments using the same algorithms in GRMS. Additional work will include a comparison of algorithms' behavior on synthetic and real workloads. All this will be available from the GSSIM portal, which will be regularly updated.

Acknowledgment

The authors would like to thank Marcin Krystek, Jakub Milkiewicz, and Stanislaw Szczepanowski for their contribution to GSSIM development. Special thanks go to Rajkumar Buyya and Michael Russell for their remarks and valuable comments.

References

[1] 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 2002 **14(13-15)**, 1175-1220 (2002).

[2] Parallel Workload Archive. http://www.cs.huji.ac.il/labs/parallel/workload/

[3] Grid Workloads Archive. http://gwa.ewi.tudelft.nl/

[4] A. Sulistio, C. S. Yeo and R. Buyya, *A Taxonomy of Computer-based Simulation and its Mapping to Parallel and Distributed Systems Simulation Tools,* Software – Practive And Experience, 2004, John Wiley and Sons, 2004.

[5] K. Aida, A. Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi and U. Nagashima, *Performance Evaluation Model for Scheduling in a Global Computing System*, Int. J. of High Performance Computing Applications **14(3)**, 268-279 (2000).

[6] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura and A. Chien, *The MicroGrid: A scientific tool for modeling computational Grids*. IEEE Supercomputing (SC2000), Dallas, TX, 4-10 November 2000. IEEE Computer Society Press: Los Alamitos, CA, 2000.

[7] A. Legrand, L. Marchal and H. Casanova, *Scheduling distributed applications: The SimGrid simulation framework.* In Proceedings 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2003), Tokyo, Japan, 12-15 May 2003. IEEE Computer Society Press: Los Alamitos, CA, 2003.

[8] W. Kreutzer, J. Hopkins and M. Mierlo, *SimJAVA – A Framework for modeling queueing networks in Java*, Proceedings of the 1997 Winter Simulation Conference ed, 1997.

[9] A. Sulistio and R. Buyya, *A Grid simulation infrastructure supporting advance reservation*. In Proceedings 16th International Conference on Parallel and Distributed Computing and Systems, Cambridge, USA, November 9–11, 2004.

[10] D. G. Feitelson, *Packing schemes for gang scheduling*. In Job Scheduling Strategies for Parallel Processing. D. G. Feitelson and L. Rudolph (eds.), pp. 89-110, Springer-Verlag 1996, Lecture Notes Computer Science vol. 1162.

[11] U. Lublin and D. G. Feitelson, *The workload on parallel supercomputers: modeling the characteristics of rigid jobs*. J. Parallel and Distributed Comput. **63(11)**, 1105-1122, 2003.

[12] M. Lo, J. Mache and K. J. Windisch, *A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling*, Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes In Computer Science **1459**, 25-46 (1998).

[13] K. Kurowski, B. Ludwiczak, J. Nabrzyski, A. Oleksiak and J. Pukacki, *Improving Grid Level Throughput Using Job Migration and Rescheduling Techniques in GRMS,* Scientific Programming, IOS Press. Amsterdam The Netherlands 263-273 (2004).

[14] J. Nabrzyski, J. Schopf and J. Weglarz, (eds.), *Grid Resource Management,* Kluwer Academic Publishers, Boston/-Dordrecht/London, 2003.

[15] The Grid Scheduling Simulations Portal, http://www.gssim.org

KRZYSZTOF KUROWSKI received the Bachelor of Science Degree from Computer Science Institute of the Poznan University of Technology in 1999. He continued his education at the Laboratory of Intelligent Decision Support System and did a second faculty at the Management and Logistic Institute. His Master Degree Diploma has been approved and honoured by the University of Paris-Dauphine in France. He joined in research and development activities conducted at Poznan Supercomputing and Networking Centre in 1999. He has taken an active interest especially in the following research fields: multi-objective optimisation, scheduling and resource management in grid environments. He has been involved in many national grid-related projects, e.g. PROGRESS, SGIGrid and Clusterix. He has taken also an active part in some EU Projects within 5 and 6 Framework Programme, e.g. Enacts, GridLab, inteliGrid and HPC-Europa. Results of his research efforts have been successfully presented at many international workshops and conferences, including IFORS, CCGrid, HPDC, Supercomputing and GGF.

JAREK NABRZYSKI received his M.Sc. and Ph.D. degrees in computer science from Poznań University of Technology in POLAND. Currently he is a researcher at the Poznań Supercomputing and Networking Center (PSNC), where he heads the Applications Department. His research interests over the last 10 years have focused on knowledge-based multiobjective project scheduling, and resource management for parallel and distributed computing. For the last couple of years he has been working on tools and middleware technologies for computational grids. Jarek Nabrzyski is a co-founder of the European Grid Forum and the Global Grid Forum. In 2001-2002 he was a member of the Global Grid Forum Steering Group where he was the Area Director of the Applications, Programming Models and Environments Area. In 2002-2005 he managed the European GridLab project, in which he was one of the Principal Investigators, responsible for such areas as Resource Management, Security and Mobile User Support. He is also involved in a number of 6FP projects, including *e.g.* ACGT, InteliGrid, GridCoord, BREIN, QosCosGrid, Challengers, BeInGrid, OMII-Europe. Jarek Nabrzyski is a member of several advisory boards, including projects such as Akogrimo, CoreGrid and UCoMs (USA). He is also a member of the KISTI Supercomputing Center Advisory Board (Korea).

ARIEL OLEKSIAK received his diploma in Computer Science at the Poznań University of Technology (Laboratory of Intelligent Decision Support Systems) in 2001. Since 2002 he has been working at the Application Department of the Poznań Supercomputing and Networking Center. His research interests include mainly resource management in Grids, scheduling, multi-criteria decision support, optimization, and knowledge discovery. He has participated in many Polish as well as international research projects related to Grids such as GridLab, SGigrid, Clusterix, HPC-Europa, and BREIN. He presented his results in many international conferences, workshops, and scientific journals.

JAN WEGLARZ, Academician, Professor (Ph.D. 1974, Dr. Habil. 1977), in years 1978-83 Associate Professor and then Professor in the Institute of Computing Science, Poznań University of Technology, member of the Polish Academy of Sciences (PAS), Director of the Institute of Computing Science, Poznań University of Technology and its predecessors since 1987, Director of Poznań Supercomputing and Networking Center, President and Scientific Secretary of the Poznań Branch of the PAS, vice President of the Committee for Computer Science of the PAS, member of the State Committee for Scientific Research, Principal Editor of the Foundations of Computing and Decision Sciences, member of several editorial boards, among others Internat. Trans. Opnl. Res. and European J. Opnl. Res. Representative of Poland in the Board of Representatives of IFORS and in EURO Council (President of EURO in years 1997-98). Member of several professional and scientific societies, among others the American Mathematical Society and the Operations Research Society of America. Author and co-author of 11 monographs, 3 textbooks (3 editions each) and over 200 papers in major professional journals and conference proceedings. Frequent visitor in major research centers in Europe and in the USA. Co-laureate of the State Award (1988) and the EURO Gold Medal (1991), laureate of the Foundation for Polish Science Award (2000).