2009 International Conference on Advanced Information Networking and Applications Workshops

A Static Benchmarking for Grid Scheduling Problems

Fatos Xhafa
Department of Languages and Informatics Systems
Technical University of Catalonia
Campus Nord, C/Jordi Girona 1-3, 08034 Barcelona, Spain
fatos@lsi.upc.edu

Leonard Barolli
Department of Information and Communication Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka 811-0295, Japan
barolli@fit.ac.jp

Juan Antonio Gonzalez
Department of Languages and Informatics Systems
Technical University of Catalonia
Campus Nord, C/Jordi Girona 1-3, 08034 Barcelona, Spain
jags@lsi.upc.edu

Pawel Jura
University of Bielsko-Biala
Bielsko-Biala, Poland
jura_pawel@wp.pl

Abstract

Analysis of algorithms for Grid computing systems before deployment in real Grid infrastructures is an important issue in Grid computing domain. Due to the complexity of real Grid systems, assessing performance analysis of optimization algorithms such as scheduling algorithms, is in general difficult, costly and time consuming. Benchmarking and simulation are two most used alternatives for analyzing optimization algorithms in Grid systems before deployment. In this paper we present a static benchmarking for scheduling problems in Grid systems. The benchmarking has been generated using the HyperSim-G Grid simulator and captures several types of Grid systems based on combinations of different machine and task types. Instances have six different sizes ranging from tiny (32 machines/512 tasks) to extra large size (1024 machines/16384 tasks) and are grouped according to machine and task types. The benchmark suite, consisting of about 720 instances, is offered through a web page.

1. Introduction and Motivation

Benchmarking is one of the most employed methodologies in computer science, engineering, business and other fields to assess the performance of important parameters of the system under study. By applying benchmarking methodology the researchers design benchmarks, also referred to as data instances or testbeds, which are used for the empirical study of the parameters of the system of interest.

Benchmarking has played an important role in computer science and combinatorial optimization to conduct performance analysis of optimization algorithms. In fact, for most well-known NP-hard optimization problems, there have been designed benchmarks that are used to compare the performance of resolution algorithms. In particular, benchmarking has been considered also as a means for the evaluation of scheduling algorithms. Taillard [14] proposed a benchmarks for basic scheduling problems. Davidovic and Crainic [5] designed a benchmark for static scheduling of task graphs with communication delays on homogeneous multiprocessor systems. Barbulescu et al. [1] presented a benchmark of test instances of scheduling for a competitive evaluation of different approaches. Watson et al. [18] consider the structured benchmarks for scheduling problems for a more realistic evaluation of algorithms, in contrast to fully random instance generation. In fact, real-world problems, such as scheduling problems in Grid systems, exhibit some proper structure and characteristics, and thus the performance of algorithms on random instances not necessarily holds on more realistic, structured instances. Other examples of benchmarks for scheduling related problems are found in [6].

978-0-7695-3639-2/09 $25.00 © 2009 IEEE
DOI 10.1109/WAINA.2009.30

It is thus generally accepted that the utilization of standard benchmark instances is essential for fair and correct comparison and analysis of heuristic algorithms. There are not, however, such standard benchmarks for several important classes of scheduling problems. This is the case of Grid scheduling, a recent class of problems that arise in the context of Grid systems. Even though scheduling in Grid systems shares several characteristics (e.g. objective functions to optimize, restrictions, etc.), with basic scheduling problems, due to the intrinsic characteristics of Grid systems [8], Grid scheduling have many proper features. Therefore, known benchmarks for scheduling problems from manufacturing, planning, multiprocessor scheduling, etc. cannot be used for the evaluation of algorithms for Grid scheduling.

Although the ultimate goal in studying the performance of scheduling methods is to use realistic dynamic Grid systems, the use of static benchmarks is important for many reasons. Static benchmarks are useful to conduct preliminary studies on scheduling methods of interest without needing to set up a Grid environment. Moreover, scheduling methods such as heuristics, usually require the calibration of a many parameters. The set of instances could be very helpful to set up the values of parameters as opposed to a blind or random assignment of parameter values. Also, when comparatively studying the performance of scheduling methods, the static benchmark is useful to repeat the experimental study many times in order to obtain relevant statistical results, which is more difficult to accomplish in real Grid systems.

In this work we present a benchmark of instances for scheduling problem on Grid systems. In designing the benchmark, we carefully took into account features of the problem that will make the benchmark useful not only for evaluation of methods in an academic context but also for scheduling in real Grid systems, in which the large-scale is a distinguished feature. The objective is thus to face the algorithms for scheduling with challenges of the problem in a real setting. Besides the size of the instances of the benchmark, we considered types of machines for expressing heterogeneity and computing capacity of Grid systems and types of tasks for evaluating scheduling techniques under different Grid scenarios. Thus, our design aims to obtain a structured benchmark of instances and captures several types of Grid systems based on combinations of different machine and task types as opposed to a purely random generation of instances.

The rest of the paper is organized as follows. We briefly review in Section 2 some related work. The design and most important features of the benchmark are presented in Section 3. In Section 4, we present some extensions of the HyperSim-G Grid simulator that we used for generating the benchmark. We describe the benchmark in Section 5 and end the paper in Section 6 with some conclusions.

2. Related Work

There have been developed many benchmarks for parallel and distributed systems but Grid system architectures are more complex than traditional systems. With the widespread use of the Grid systems in many fields, especially for the eScience problems, researchers have started to develop benchmarks to specifically match the needs of evaluations of Grid systems. Nonetheless, most of these efforts are oriented towards benchmarks for Grid systems in general; benchmarks specifically designed for scheduling problems in Grids are lacking.

Snavely et al. [12] reviewed benchmarks for grid computing in the literature. The authors stressed the importance to deploy measurement methods for Grid applications and architectures as well as the benefits of Grid benchmarks to the study and design of Grids. The authors conclude that it is imperative to develop and deploy sets of Grid benchmarks.

Venkata Rao et al. [16] proposed a set of low-level grid benchmarks to evaluate a series of grid parameters such as middleware overheads due to inter-site communication for different message sizes, data transfer, basic rates of networks, and grid information service. The benchmark seems suitable to measure the optimization of the performance of the date analysis process in a grid application. In particular, using the benchmark, the overheads incurred by Grid Resource Broker for task submission, and the efficiency of scheduling algorithms for different scenarios can be analyzed.

Frumkin and Van der Wijngaart [9] presented NAS Grid Benchmarks (NGB), based on NAS Parallel Benchmarks. NGB is conceived as a data flow graph encapsulating an instance of an NAS Parallel Benchmark code in each graph node, which communicates with other nodes by sending/receiving initialization data.

An interesting initiative is that of the Grid Workloads Archive [10], motivated also by parallel workload archives and parallel production environments. A database of existing grid workload traces as well as tools that convert workload trace formats specific to different platforms are offered. It is said that workload generation tools are able to generate workloads that meet user-specified criteria (e.g. system load).

Chun et al. [4] developed a set of probes for measuring the performance and failure rates of basic grid operations. These measurements include compute times, network transfer times, and Globus middleware overhead. Tsouloupas and Dikaiakos [15] presented a core set of benchmarks for characterizing Grid nodes or collections of Grid resources. A framework for running benchmarks on Grid environments is also provided. Jawari [11] is an extensible, multiplatform grid benchmarking service that provides the grid community with support to assess and improve the quality of service delivered by grid infrastructures. The benchmarking service can be used, for instance, to identify performance bottlenecks.

There have been also other works in this research line, although closer to simulation needs rather than benchmarking. Flavihe et al. [7] presented a simulation framework for the Semantic Grid Environment; the authors used GridSim simulation library [3]. Sulistio et al. [13] used the GridSim for simulating data grids. The framework is especially suited for simulating data intensive tasks.

3. The Benchmark Design

The objective of the benchmark for Grid scheduling problem is to reveal the performance of algorithms and heuristic methods for solving the problem. It is thus necessary, on the one hand to carefully choose the model and the parameters that will determine the generation of the instances, and on the other, to abstract features of the problem from real Grid environments so that the generated instances be representative for the type of challenges Grid schedulers may face.

3.1 The model for generating instances

We use the Expected Time to Compute (ETC) model [2] in which, the instance is a matrix ETC of size $nb\_machines \times nb\_tasks$, each position $ETC[t][m]$ of which indicates the expected time to compute task t on machine m. A benchmark of instances for distributed heterogenous systems is generated in this model [2] by combining three characteristics: computing consistency (consistent / semi-consistent / inconsistent), heterogeneity of resources (low / high) and heterogeneity of tasks (low / high).

An ETC matrix is considered consistent when, if a machine m_i executes task t faster than machine m_j, then m_i executes all the tasks faster than m_j. Inconsistency means that a machine is faster for some tasks and slower for some others. An ETC matrix is considered semi-consistent if it contains a consistent sub-matrix.

An instance in the ETC benchmark is just an ETC matrix, randomly generated, which satisfies a combination of computing consistency, heterogeneity of resources and heterogeneity of tasks.

It should be noted that using only the ETC matrix information could be insufficient for modelling the system. Indeed, in many Grid scenarios the explicit information of workloads of tasks and information on computing capacity of machines is necessary. For instance, some heuristic methods may take as input the vector of task workloads (e.g. expressed in millions of instructions) and the vector of computing capacity of machines (e.g. in Mips –millions of instructions per second). As an example, the LJFR-SJFR (*Longest Job to Fastest Resource - Shortest Job to Fastest Resource*) heuristic works with the information on tasks and machines in an explicit way.

In fact, disposing of the information on task workloads and computing capacity of machines allows to compute the ETC matrix by simply dividing, for any task t and machine m, the workload of task t by computing capacity of machine m.

The computation of task heterogeneity, machine heterogeneity and consistency of computing characteristics can be done as follows.

Task heterogeneity. This characteristic can be easily identified based on the variance of the data of the vector of task workloads. Then, based on a threshold value (*a priori* fixed) tasks can be classified as of high or low heterogeneity. Assuming that there are nb_tasks, the computational cost of deciding task heterogeneity is $O(nb\_tasks)$.

Machine heterogeneity. This characteristic is identified similarly as in the case of task heterogeneity but now the vector of computing capacity of machines is used. Again, based on the variance and a (different) threshold value, machines are classified as of high or low heterogeneity. Assuming that there are $nb\_machines$, the computational cost of deciding task heterogeneity is $O(nb\_machines)$.

Consistency of computing. It is a bit more complex and computationally expensive to identify the computing consistency of the ETC matrix. Recall that, the matrix falls into one of the following three cases: consistent, semi-consistent or inconsistent. This is done by exploring the ETC matrix in a way that for a given machine m_i, we scan the set of machines m_j, $i < j$ and comparing for each task the corresponding ETC values for machines i and j. The overall computation in this case is $O(nb\_machines^2 \cdot nb\_tasks)$.

3.2 Parameters for generating benchmark instances

In generating the instances of the benchmark we considered the following three parameters: instance size, types of tasks and types of machines.

Instance size. Any benchmark should consider a set of instances with varying sizes of the data. In the case of Grid scheduling, the size of the instances is a crucial parameter. In this respect, the benchmark of Braun et al. [2] is limited given that all instances consists of 512 tasks and 16 machines. Computational Grids could much vary in their size; e.g. Campus Grids are composed of a few clusters while planetary-scale Grids virtually join up to thousands of machines[1]. Moreover, the potential number of users submitting tasks to a Grid system is expected to be large. It is therefore important to consider instances ranging in size from *tiny* (a dozens of machines and hundreds of tasks) to *large* and *very large* (thousands of machines and thousands of tasks). Instance size is also relevant for studying the scalability of scheduling algorithms. By running the scheduling algorithms on instances of varying sizes, it is possible to deduce how well the algorithms under study scale. Such studies would be helpful to deduce whether the Grid scheduler based on the chosen algorithm should be centralized or decentralized.

Number of instances per size. Given an instance size, the benchmark should contain many different instances of that size. In this way, it is possible to run scheduling algorithms in different instances of the same size and evaluate their performance. A scheduling method that shows the same performance pattern for most of the instances could be considered robust and is expected to show the same pattern of performance for other similar instances.

Types of tasks. Types of tasks are important for evaluating scheduling techniques under different Grid scenarios. Generating different workloads of task is among most important aspects of the benchmark. Indeed, pure random generation of workloads cannot capture the nature of different types of tasks, as regards their workload, in Grid systems. Tasks submitted to Grid systems could vary a lot in their computing needs. Some of them may be originated from scientific and engineering computations and others from financial and business applications. Recently, two well distinguished classes of tasks are the so-called *computing intensive* and *data intensive*.

A computing intensive task is any task that is speed limited by how fast the processor can compute the data; such tasks require a lot of computations. Computing intensive tasks are usually complex computations even they could handle small datasets, the consumption in this case is the CPU. Data intensive tasks, consists of processing massive datasets of Gigabytes or Terabytes, and extract relevant information from the data. Such datasets arise from many application domains, including eScience problems (e.g. bioinformatics, weather forecast, etc.), security (e.g., processing large server log files), commerce (e.g. mining large datasets of clients' records). These two types of tasks[2] are thus interesting for measuring the performance of Grid schedulers. Our model is inspired by the IBM's Compute Grid programming, which includes batch and compute-intensive models within the WebSphere Software.

Types of machines. Machines in a Grid systems could vary not only as regards their computing capacity but also their "specialization", that means, some machines could be *specialized* for numerical computations, fast I/O operations. Therefore, besides the heterogeneity of machines in terms of the computing capacity (fast *vs.* slow CPU), other types of machines can be considered in order to more realistically match Grid scenarios.

4. Extension of the HyperSim-G Simulator for Generating the Benchmark

For generating the benchmark we have used a Grid simulator, namely HyperSim-G, a discrete event-based simulator by Xhafa et al. [17]. The simulator has been conceived, designed and implemented as a tool for experimentally studying algorithms and applications in Grid systems. The simulator is highly configurable through the following parameters:

- `Init. hosts`: Number of resources initially in the environment.
- `Max. hosts`: Maximum number of resources in the grid system.
- `Min. hosts`: Minimum number of the resources in the grid system.

[1]PlanetLab is an example of planetary scale grid systems, currently consists of 942 nodes at 469 sites. http://www.planet-lab.org/

[2]Also referred to as immediate and batch tasks in scheduling literature.

- `Add host`: Normal distribution for time interval arrival of new resources.
- `Delete host`: Normal distribution for time interval leaving of resources.
- `Total tasks`: Number of tasks to be scheduled.
- `Init. tasks`: Initial number of tasks in the system to be scheduled.
- `Workload`: Normal distribution for the workload of tasks.
- `Interarrival`: Exponential distribution for the time interval of task arrivals.
- `Activation`: Establishes the activation policy according to an exponential distribution.
- `Reschedule`: When the scheduler is activated, this parameter indicates whether the already assigned tasks, which have not yet started their execution, will be rescheduled.
- `Host selection`: Selection policy of resources (all means that all resources are selected for scheduling purposes).
- `Task selection`: Selection policy of tasks, (all means that all tasks must be scheduled).
- `Local policy`: The local policy of scheduling tasks to resources. SPTF (Shortest Processing Time First Policy) -in each resource is executed first the task of smallest completion time.
- `Number runs`: Number of simulations done using the same parameter configuration.

In order to include the type of machines and types of tasks, the simulator in [17] has been extended with new functions and parameters:

- `Threshold of machine type`: Expresses the classification of machines into two types.
- `Threshold of task type`: Expresses the classification of tasks into two types.

These parameters are indicated to the simulator through the following options:

```
"-3, --thrmach <double> Threshold of machine type"

"-4, --thrtask <double> Threshold of task type"
```

It should be noted that in order to generate a static benchmark of instances, we used the *same values* for parameters related to the number of hosts (Initial number of hosts, Minimum number of hosts, Maximum number of hosts) and the *same values* for parameters related to the number of tasks (Total number of tasks and Initial number of tasks).

5. Benchmark Description

The benchmark is generated using HyperSim-G simulator by combining the size of instances, types of tasks and types of machines within one instance size. The combination of these characteristics allows to more realistically measuring the performance of scheduling techniques. The following terms and notations are used for description of benchmarks. Notice that the benchmark includes both instances and simulator traces from which they were extracted.

Instance sizes. We have considered six instance sizes ranging from tiny to extra very large size:

- t- tiny: 32 hosts/512 tasks.
- s- small: 64 hosts/1024 tasks.
- m- medium: 128 hosts/2048 tasks.
- l -large: 256 hosts/4096 tasks.
- xl- very large: 512 hosts/8192 tasks.
- xxl- extra very large: 1024 hosts/16384 tasks.

For each size, we have generated a total of 121 instances, which are grouped into 11 groups, each having 11 instances. This grouping is done by the combination of machine and task types (e.g. A 20% | B 80% - for machines means that 20% of the machines are of type A (specialized for numerical computations) and 80% are of type B). Similarly for tasks, A 40% | B 60% - for tasks means that 40% of the tasks are of type A (computing intensive) and 60% are of type B (data intensive).

Instance format. We give below (see Table 1) the format of the instance, that is, the order in which the data for tasks, machines and ETC matrix are arranged inside the instance.

Note that the vector of ready times is omitted in the instances, actually ready times are all set to 0 (all machines are available).

Table 1. Instance format

Instance format	Explanation
#tasks	Number of tasks
#machines	Number of machines
workload T1 ... workload TT	Vector of task workloads (integer values)
MIPS M1 ... MIPS MM	Vector of computing capacities of machines (integer values)
ETC[T1][M1]...ETC[T1][MM] ETC[T2][M1]...ETC[T2][MM] ... ... ETC[TT][M1]...ETC[TT][MM]	ETC values by rows starting with first task on all machines and so on up to the last task.

Output Trace. All instances are associated with the corresponding output trace of the simulator from which they are extracted.

Web interface for the benchmark The benchmark suite is available through a web site at: http://weboptserv.lsi.upc.edu/WEBGRID/ (Benchmark section).

6. Conclusions and Future Work

In this work we have presented a static benchmarking for scheduling problems in Grid systems. The benchmarking has been generated using the HyperSim-G Grid simulator and captures several types of Grid systems based on combinations of different machine and task types. Instances have six different sizes ranging from tiny (32 machines/512 tasks) to extra large size (1024 machines/16384 tasks) and are grouped according to machine and task types. The benchmarking, consisting of about 720 instances, is offered through a web page and can be freely downloaded.

We are currently extending the web interface to allow remote users to generate their own static instances.

References

[1] L. Barbulescu, L. Kramer, and S. Smith. Benchmark problems for oversubscribed scheduling. *Workshop on Scheduling Competition, The 17th International Conference on Automated Planning & Scheduling* , 2007.

[2] T. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, and B. Yao. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6):810–837 (2001).

[3] R. Buyya and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurr. Comput. : Pract. Exper.*, 45(14):1315, 2002.

[4] G. Chun, H. Dail, H. Casanova, A. Snavely. Benchmark probes for grid assessment. *Proceedings of 18th International Parallel and Distributed Processing Symposium*, 276-284, 2004

[5] T. Davidović and T. G. Crainic. Benchmark-problem instances for static scheduling of task graphs with communication delays on homogeneous multiprocessor systems *Computers and Operation Research*, 33(8),2155–2177, 2006.

[6] Benchmarks and examples. http://www.faqs.org/faqs/constraints-faq/part1/section-9.html

[7] A. Flahive, W. Rahayu, D. Taniar and B. Apduhan. A Distributed Ontology Framework in the Semantic Grid Environment, In *AINA'05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, 193–196, IEEE CS, 2005.

[8] I. Foster and C. Kesselman. *The Grid - Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[9] M. Frumkin and R.F. Van der Wijngaart NAS Grid Benchmarks: A Tool for Grid Space Exploration *Cluster Computing*, 5(3), 247–255, 2002.

[10] The Grid Workloads Archive http://gwa.ewi.tudelft.nl/pmwiki/

[11] The Jawari benchmark suite. http://www.itwm.fraunhofer.de/de/hpc__Jawari/Jawari/

[12] A. Snavely, G. Chun, H. Casanova, R.F. Van der Wijngaart and M. Frumkin. Benchmarks for grid computing: a review of ongoing efforts and future directions. *SIGMETRICS Perform. Eval. Rev.*, 30(4), 27–32, ACM, USA, 2003.

[13] A. Sulistio, U. Cibej, S. Venugopal, B. Robic and R. Buyya. A toolkit for modelling and simulating data Grids: an extension to GridSim. *Concurr. Comput. : Pract. Exper.*, 20(13), 1591–1609, 2008.

[14] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278-285,1993

[15] G. Tsouloupas and M.D. Dikaiakos. GridBench: A Tool for Benchmarking Grids. In *Proceedings of the 4th International Workshop on Grid Computing*, 60-66, IEEE CS, 2003

[16] V. Rao, B.R. Swami. Low-Level Grid Computing Benchmarks *International Conference on High Performance Computing*, 2005.

[17] F. Xhafa, J. Carretero, L. Barolli and A. Durresi. Requirements for an Event-Based Simulation Package for Grid Systems. *Journal of Interconnection Networks*, 8(2):163-178, 2007.

[18] J.P. Watson , L. Barbulescu , L.D. Whitley , and A.E. Howe. Contrasting Structured and Random Permutation Flow-Shop Scheduling Problems: Search-Space Topology and Algorithm Performance. *INFORMS Journal on Computing*, Vol. 14, No. 2, Spring 2002, pp. 98-123