A Simulation Model for Grid Scheduling Analysis and Optimization

Florin Pop Ciprian Dobre Gavril Godza Valentin Cristea

Computer Science Departament, University "*Politehnica*" of Bucharest, Romania
{florinpop, cipsm, gavrilg, valentin}@cs.pub.ro

Abstract

Scheduling is an important research topic in Grid computing. This is due to the aim of Grids to offer high quality services to members of Virtual Organizations based on the efficient use of the available resources. This goal can be achieved through a good scheduling strategy applied to the local lever (clusters) and global level (entire system) of Grids. Since the scheduling problem is NP complete, we can afford only sub-optimal solutions to this problem. In addition, the highly dynamic behavior of Grid components (users, processes, resources) makes more difficult the finding of a good scheduling solution. In this paper we propose a new approach to solving the scheduling problem by simulation. Using a simulator has the merit to shorten the distance between the real Grid system and the model used for its analysis. In order to transform the simulator into a useful tool able to cope with the Grid dynamicity, we propose a solution that couples the simulator with a Grid monitoring / optimization tool, so that scheduling decisions are taken and used in real time for the next short period.

Keywords

Simulation, Scheduling, Grid computing, Virtual Organizations, MONARC 2.

1. Introduction

An important research issue in Grid computing is scheduling. Obtaining significant results in this area will contribute to the success of Grid platforms in satisfying the demands of users by efficiently exploiting the resources made available by the Grid service providers. Scheduling algorithms proved to be important in "conventional" systems, and have been studied in research activities targeted to ensure the optimal behaviour of these systems from several points of view, such as the response time and the resopurce use. The results obtained in this research cannot be applied to the Grid, which brings some new functional requirements and objectives. First, the Grid resources are very heterogenous and their availability is not predictable. Then, the scheduling decisions are taken dynamically based on user requirements. The scheduling algorithms should be based on zero knowledge of the Grid structure and actual resources. In the same time, they should consider the common features of Grid platforms, which derive from the concept of Virtual Organizations (VOs) that induce two basic principles of Grid operation: resources belonging to different user organizations are used for solving a common problem; the same resources are used also by people outside the VO.

Several schedulers for grid computing systems have been developed. We mention here Nimrod-G that is an economy based scheduler, GRaDS that aims maintain high performance by adapting the application according to changes in the available resources, and Condor-G that is a task broker designed to front end a computational grid. The model used in the Grid scheduling systems considers that job allocation is done in two stages. Firstly a job is allocated to a particular node on the grid and then within that node, the job will be scheduled onto the processor. In order to share Grid resources, the scheduler should allow the allocation of more than one machine for a single aplication, and the allocation of more applications on the same machine.

The problem of scheduling, and implicitely of the evaluation of scheduling algorithms, is NP-complete. An approach based on analytic solution or on the exhaustive exploration of the solution space is not feasible. Instead, the evaluation by simulation is a practical and acceptable solution adopted also in other studies. Our approach was to make an extension to an existing simulator in several directions: (1) to facilitate the evaluation of scheduling algorithms by the inclusion of their descriptions as an input of the simulator, (2) to al-

Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)
0-7695-2554-7/06 $20.00 © 2006 IEEE

low the evaluation of the scheduling algorithms in realistic circumstancies by feeding the simulator with monitoring data collected during Grid functionning, (3) to determine the best scheduling policy for the next short time period and provide it as a feedback to the Grid platform.

In this paper we describe this approach and the results obtained so far. The structure of the paper is as follows. Section 2 presents some characteristics of simulators for distributed systems and how they are considered in some existing simulator tools. Section 3 describes some challenges concerning the scheduling algorithms in Grid computing. Our solution and some partial results obtained so far are presented in Section 4. Finally, in Section 5 some conclusions and future work are presented.

2. Simulating distributed systems

Simulation is a powerful method to perform system analysis, even when the involved system does not physically exist or if it is very expensive to use it. Briefly, one must design a system model, as close to the real system as possible, and then run the model in order to collect results.

Running a model means to monitor the system over a certain period of virtual (simulation) time. This simulation time can flow either continuously (for systems described using differential equations) or discrete (for event based systems). We will focus on discrete event simulation, because this is suitable for computer systems. The steps to be followed when simulating a systems are:

- Identify the simulation frame and goals;
- Create the conceptual model;
- Define simulation experiments;
- Collect input data and prepare system's parameters;
- Create the simulation model;
- Verify and validate the simulation model;
- Run simulation experiments;
- Collect simulation results;
- Analyse the simulation results.

Each step has its own importance, but defining the simulation experiments seems to be the core of the entire process. After start-up, the system enters a loop, till the stop condition is fulfilled. At each step the system state is updated and the time advances. Of course, updating the system state means several operations, including event scheduling. As a side effect, relevant simulation information is collected.

There are several approaches to follow these steps. One can design from scratch the entire process, but this is not very efficient. Of course a general system simulator can be used, but the best solution is to use a dedicated simulator.

Simulators tools for distributed systems

Various Grid simulation projects have been undertaken in recent years, among them MONARC, OptorSim, ChicagoSim, EDGSim and GridSim. This section briefly describes these simulation tools by highlighting their key similarities and differences.

OptorSim is a simulator project developed by a team of researchers working on the GridPP project. The GridPP is a collaboration of particle physicists and computing scientists from UK and CERN, who are building a Grid for particle physics. The main advantage of OptorSim with respect to the other simulators described is that it performs two-stage optimization. Scheduling decisions are based on both the location of data and the status of network links between grid sites, while (re)optimization during the run-time of a job takes into account dynamic variations in the distribution of data and in the behavior of network resources.

ChicagoSim is a simulator developed by a team of researchers from The University of Chicago. The simulator is modular and extensible and uses Parsec, a discrete event simulation tool to model events like file requests and data transfers. ChicagoSim targets systems consisting of potentially large numbers of resources, resource owners, and VOs.

EDGSim was designed to simulate the performance of the EU DataGrid but concentrates on the optimisation of scheduling algorithms. It provides a simulation of the flow of computational jobs, physics data and information around a computational grid based on the software being developed by the European Data Grid project. In EDGSim each entity in the virtual grid is represented by one of these Actor objects, and any kind of communication, whether it is a message passed by the Information Services or a job being submitted, is represented by a Data Token. There are some interesting simulations done using EDGSim. One such simulation showed that data location was important in the scheduling decision, but during that particular simulation it is interesting to note that no replication of data was taken into account. A similar simulation was done

using MONARC, but the data replication was taken into account which led to somewhat different results.

GridSim is a discrete event simulator written in Java developed by researchers from the Gridbus project. The simulator is based on SimJava, which is a process based discrete event simulation package for Java with animation facilities. GridSim supports modeling and simulation of heterogeneous Grid resources (both time- and space-shared), users, applications, brokers and schedulers in a Grid computing environment. It provides primitives for creation of application tasks, mapping of tasks to resources, and their management so that resource schedulers can be simulated to study the scheduling algorithms involved. The GridSim project is very detailed in its simulation of the components of a Grid and introduces an economic model to manage the use of Grid resources through the buying and selling of resources. It is designed primarily to study scheduling algorithms and does not examine the issue of data replication. Beside that limitation GridSim also does not support advance reservation, does not offer realistic results in relation to moving or the storage of data, the network functionality is very limited and also results output is crude.

MONARC 2 is a simulator which is developed by a mixt team of researchers (Caltech-CERN-PUB). The main goal of the MONARC [1][1] project is to provide a realistic simulation of large distributed computing systems and to offer a flexible and dynamic environment to evaluate the performance of a range of possible data processing architectures. To achieve this purpose, the simulator provides the mechanisms to describe concurrent network traffic and to evaluate different strategies in data replication or in the job scheduling procedures. Building the logical simulation model requires the abstraction from the real system for all the components and their time dependent interaction. This logical model has to be equivalent to the simulated system in all important respects [1].

From the beginning MONARC 2 simulator project was designed starting from the fact that the scale, complexity and worldwide geographical spread of the LHC (Large Hadron Collider) computing and data analysis problems are unprecedented in scientific research. The complexity of processing and accessing this data is increased substantially by the size and global span of the major experiments, combined with the limited wide area network bandwidth available. The simulation program requires the abstraction of all components from the real systems and their time dependent interactions. This abstracted model has to be equivalent to the original system in the key respects that concerns us. The simulation engine is designed to be generic for any distributed systems. However, there are certain HEP-specific system components that are specially modeled to make the tool useful to the physics community. With the structure that MONARC 2 provides for Grid computing simulation is possible to build a wide range of computing models, from the very centralized (with reconstruction and most analyses at CERN) to the distributed systems, with an almost arbitrary level of complication (CERN and multiple regional centers, each with different hardware configuration and possibly different sets of data replicated) [7].

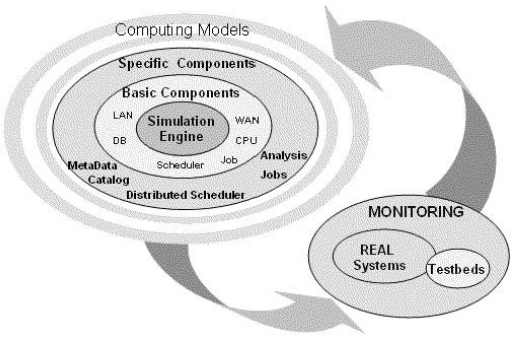

Figure 1. MONARC2 Architecture

When comparing MONARC 2 with other existing simulator tools as the ones described earlier one can see a number of points were MONARC 2 had a clear advantage. MONARC 2 is the most flexible and extensible application there is. The simulator uses an Object Oriented design, which allows an easy and direct mapping of the logical components into the simulation program and provides the interaction mechanism, offering the best solution for such a large scale system even if the system changes dynamically. Each simulated entity in MONARC 2 can be modified according to the simulated scenario, but also new entities can be easily added into the simulated scenario. The simulation can be flexible modeled at different level of abstraction. MONARC 2 provides a general simulation framework, one that can also be used in domains other than physics experiments.

This simulator uses the Regional Center model and provides a very realistic way to simultate all the elements that are parts of a distributed system (node, CPU, LAN, link, protocol, storage, replication, scheduling, etc.). Due to its layered structure MONARC 2 can be easily extended, even by the user. The first two layers contain the core of the simulator (the simulation engine) and models the basic components of a distributed system, being the foundation for other particular components, specific to the simulated

[1] **MO**dels of **N**etworked **A**nalysis at **R**egional **C**enters

systems. The simulation engine is continuously tested and improved in order to provide a clear advantage in terms of performance and also to provide the possibility to work with a large number of threads. The simulated data models are generals and the tool can be used to simulate a wide range of data replication algorithms or the simulation of distributed scheduling algorithms. Also MONARC 2 provides realistic modeling of local area and large area networks and offers a large set of methods for data visualization through its graphical user interface.

3 Challenges of Scheduling Algorithms in Grid Computing

Grid scheduling can be defined as the process of making efficient scheduling decisions. In scheduling process are involving resources over multiple administrative domains. It is possible to search multiple administrative domains to use a single machine or scheduling a single job to use multiple resources at a single site or multiple sites.

More applications are turning to Grid computing. Applications need to meet their computational and data storage needs. Single sites are simply no longer efficient for meeting the resource needs of high-end applications, and using distributed resources can give the application many benefits. Effective Grid computing is possible, however, only if the resources are scheduled well.

In scheduling context we can define a job to be anything that needs a resource - from a bandwidth request, to an application, to a set of applications. We use the term resource to mean anything that can be scheduled: a machine, disk space and memory, processors, a QoS network [10]. Because our propose refers to simulation and results analysis, we start with evolution of scheduling algorithms with parallel and distributed computing systems. It is important to know what platform we use for scheduling simulation. Examples of machines for scheduling are parallel and distributed systems, such as symmetric multiple processor machines, massively parallel processors computers and cluster of workstations. Scheduling algorithms have been intensively studied as a basic problem on that type of machines. Scheduling algorithms are evolving with the architecture of parallel and distributed systems.

Traditional systems are described in the following way [3]: all resources reside within a single administrative domain, the resource pool is invariant, to provide a single system image, the scheduler controls all of the resources, contention caused by incoming applications can be managed by the scheduler according to some policies, so that its impact on the performance that the site can provide to each application can be well predicted, computations and their data reside in the same site or data staging is a highly predictable process, usually from a predetermined source to a predetermined destination, which can be viewed as a constant overhead [12].

We captured description of some important features of parallel and distributed systems and typical scheduling algorithms they adopt. The tabelar result are shown in 2. These information is important for our simulating platform and represent one of the keys in the evaluation of algorithms [8].

TYPICAL ARCHITECTURE	MULTIPLE PROCESSOR MACHINES	CLUSTER OF WORKSTATIONS	GRID SYSTEMS
History	Late 1970s	Late 1980s	Mid 1990s
Typical System Interconnect	Bus , Switch	Commercial LAN, ATM	WAN / Internet
Cost of Interconnection	Very Low / Negligible	Low / Not Negligible	High / Not Negligible
Interconnection Heterogeneity	None	Low	High
Node Heterogeneity	None	Low	High
Single System Image	Yes	Yes	No
Resource Pool Static/Dynamicity	Predetermined and Static	Predetermined and Static	Not Predetermined and Dynamic
Resource Management Policy	Monotone	Monotone	Diverse
Typical Scheduling Algorithms	Homogeneous Scheduling Algorithms	Heterogeneous Scheduling Algorithms	Grid Scheduling Algorithms

Figure 2. Traditional systems characteristics

There is many way to construct scheduling algorithms for grid systems. We have a hierarchical taxonomy for scheduling algorithms in general-purpose parallel and distributed computing systems. This taxonomy is simple represented in figure 3. Since Grid is a special kind of distributed systems, scheduling algorithms in Grid fall into a subset of this taxonomy.

If we analyse this taxonomy [6], from the top to the bottom, this subset can be identified as: local and global scheduler, static and dynamic scheduler.

At the highest level, a distinction is drawn between local and global scheduling methods. The **local scheduling** determines how the processes resident on a single CPU are allocated and executed. The **global scheduling** policy uses information about the system to allocate processes to multiple processors to optimize a system-wide performance objective. Obviously, Grid scheduling falls into the global scheduling branch. This is the local versus global perspective.

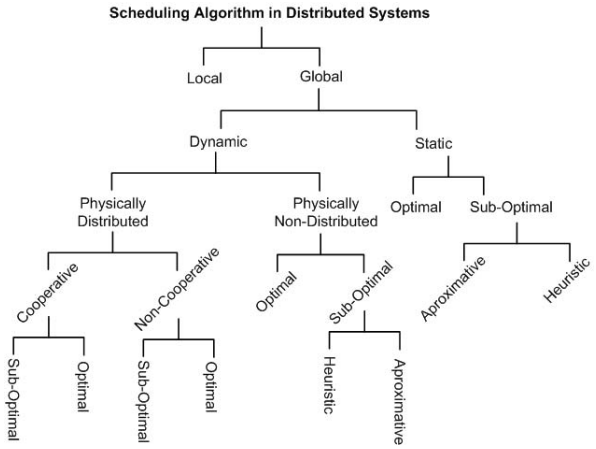

Figure 3. Taxonomy of grid schedulers

The next level in the hierarchy (in the global scheduling) is a choice between static and dynamic scheduling. This choice indicates the time at which the scheduling decisions are made. In **static scheduling** information is assumed to be available by the time the application is scheduled. This information regarding all resources in the Grid as well as all the tasks in an application. In **dynamic scheduling**, the basic idea is to perform task allocation on the fly as the application executes. This is useful when it is not possible to determine the execution time, direction of branches and number of iterations in a loop as well as in the case where jobs arrive in a real-time mode. These variances introduce forms of non-determinism into the running program. Both static and dynamic scheduling are widely adopted in Grid computing [9].

An important aspect in scheduling process and is **data scheduling**. Is important because, in high energy physics for example, and other disciplines, there are applications involving numerous, parallel tasks that both access and generate large data sets (like petabyte). Data sets in this scale require specialized storage and management systems and data Grid projects are carried out to harness geographically distributed resources for such data-intensive problems by providing remote data set storage, access management, replication services [4], and data transfer protocols [2].

It is usefull in the simulation process to have a method for this kind of problem. Simulating more algorithms we can determinate what decision we must adopt and in what situation. [11].

The new approaches for grid task scheduling consider Grid to be a kind of system made up of a large number of autonomous resource providers and consumers, which are running concurrently, changing dynamically, interacting with each other but self-ruling.

In nature and human society, there are some systems having the similar characteristics. For example economic systems. The new approaches, such as the Grid Economy and other heuristics inspired by this natural phenomena, were proposed in last years to address the challenges of Grid computing [5]. Ideas behind these approaches are not originally applied to the scheduling problem and mapping from the problem spaces in which they are initially used to Grid scheduling problems is usually required. Figure 4 present the framework of the new approaches.

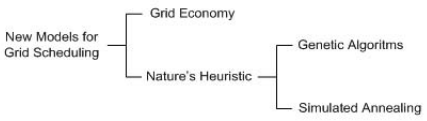

Figure 4. New approach for Grid Scheduling

Like many other new models this approach introduce some new problems in connection with the choice of objective functions and the mapping of a classic scheduling problem to the new model. New objective functions considering the economic cost of computing and their optimization are far from having been thoroughly studied. As well the economic models in Grid computing themselves need to be refined to apply to more sophisticated applications which include task dependency and require cooperation among different resource providers. And in a genetic scheduling algorithm, one open problem is how to use dynamically predicted performance information to help the speed of convergence (as the current genetic scheduling algorithms are static).

This is a simple presentation of the scheduling universe. The grid systems need a good mechanism for scheduling problem that mean finding the best policy for running applications. The scope of our simulation is to compare some type of grid scheduling algorithm on the various platforms and find the optimal solution for one platform type.

4 Solution based on simulation

The selection of the scheduling algorithm is highly dependent on a variety of constraints such as schedule restrictions given by the system, like the availability of dynamic partitioning or gang scheduling or system parameters, like I/O ports, node memory and processor types. Constraints that refer to jobs are: distribution of job parameters, like the amount of large or small jobs, availability and accuracy of job information for

the generation of the schedule. For instance, this may include job execution time as a function of allocated processors. An alternative scope is definition of the objective function.

In figure 5 is a graphic that represents the rates of submitted jobs, finished jobs, imported jobs and exported jobs (measured in jobs per hour) for the Cern regional center in the a simulation with MONARC 2[2].

This is a simple example of simulation method. Our solution propose simulation for some different type of scheduling algoritms and obtain some results. This results will be used in selection phase of algoritms for specific platform.

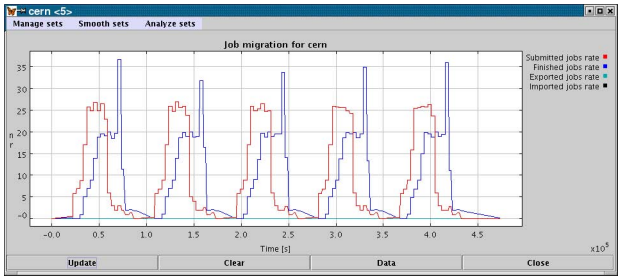

Figure 5. Simulation: Job rate statistics for the Cern center

We can compare the efficiency for different scheduling algorithms by measuring parameters such as the average time needed for processing a job or the CPU utilization. As the number of CPUs decreases, the CPU utilization is improved, but the job processing time grows since the resources are not sufficient.

5 Conclusions and future work

We propose a simulation model for drid scheduling analysis and optimization for Romanian CNCSIS project. The main objective of are the development of a model, and of an associated tool, for the dynamic evaluation of the scheduling algorithms in Grid systems based on user requirements.

We will approaches a key problem for ensuring the high performance behavior of the Grid: the scheduling of the activities. The scheduling algorithms have been also studied in research activities targeted to "conventional" systems; but the known results cannot always be applied to the Grid, which brings some new functional requirements and objectives: the resources that form the system are very heterogenous, the scheduling decisions are taken dynamically based on user requirements, no knowledge of the structure of the system is necessary in order to execute programs.

[2]see monalisa.cacr.caltech.edu/MONARC

We start with a general simulation model which includes the base components (processing units, network protocols, databases, jobs etc.). The model will be extended and adapted for the simulation of Grid systems and of the scheduling and resource management algorithms. The main future works in the project are: the analysis of the scheduling algorithms currently used in the Grid systems; the establishment of relevant performance measures and of an evaluation model; the evaluation of the current scheduling algorithms, using the chosen model; the elaboration of a proposal for optimizing the scheduling algorithms and the evaluation model.

References

[1] Monarc homepage. *http://monarc.cacr.caltech.edu*, Accessed 15th March 2006.

[2] W. Allcock. The globus striped gridftp framework and server. *Proc. of the 2005 ACM/IEEE conference on Supercomputing*, Seattle, Washington USA:54–64, November 2005.

[3] F. Berman. *High-Performance Schedulers, chapter in The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.

[4] J. Bester. Gass: A data movement and access service for wide area computing systems. *Proc. of the 6th Workshop on I/O in Parallel and Distributed Systems*, Atlanta, Georgia USA:78–88, May 1999.

[5] R. Buyya. Economic models for resource management and scheduling in grid computing. *J. of Concurrency and Computation: Practice and Experience*, Wiley Press:1507–1542, December 2002.

[6] T. Casavant. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Trans. on Software Engineering*, 14(2):141–154, February 1988.

[7] C. M. Dobre. Monarc simulation framework. *Proceedings of the RoEduNet International Conference, Timisoara, Romania*, May 2004.

[8] F. Dong. Scheduling algorithms for grid computing: State of the art and open problems. *Technical Report No. 2006-504, School of Computing, Queens University Kingston, Ontario*, January 2006.

[9] H. El-Rewini. *Task Scheduling in Parallel and Distributed Systems*. PTR Prentice Hall, 1994.

[10] J. F. D. Jongh. *Share Scheduling in Distributed Systems*. Ph. D. Thesis, Technische Universiteit Delft, 2002.

[11] S. Park. Chameleon: a resource scheduler in a data grid environment. *Proc. of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid03)*, Tokyo Japan:253–260, 2003.

[12] Y. Zhu. *A Survey on Grid Scheduling Systems, Department of Computer Science*. Hong Kong University of science and Technology, 2003.