Job Scheduling and Resource Management Techniques in Economic Grid Environments[*]

Rafael Moreno[1] and Ana B. Alonso-Conde[2]

[1] Departamento de Arquitectura de Computadores y Automática, Universidad Complutense, 28040 - Madrid, Spain
`rmoreno@dacya.ucm.es`

[2] Dept. Economía Financiera, Contabilidad y Comercialización, Univ. Rey Juan Carlos, 28032 - Madrid, Spain
`abac@fcjs.urjc.es`

Abstract. In this paper, we analyze the problem of grid resource brokering in the presence of economic information about the price of resources. We examine in detail the main tasks that a resource broker has to carry out in this particular context, like resource discovery and selection, job scheduling, job monitoring and migration, etc. Then, we propose an extension of the grid resource information service schema to deal with this kind of economic information, and we evaluate different optimization criteria for job scheduling and migration, combining both performance and economic information. The experimental application benchmark has been taken from the finance field, in particular a Monte Carlo simulation for pricing European financial options.

1 Introduction

Computational Grids are emerging as a new computing paradigm for solving grand challenge applications in science, engineering, and economics [1]. Grid development involves the efficient management of heterogeneous, geographically distributed, and dynamically available resources. In this environment, the resource broker (or scheduler) becomes one of the most critical components of the grid middleware. Following, we analyze in detail the main tasks that the resource broker has to carry out [2], and the most common approaches to perform these tasks, specially in case of the presence of economic information.

Resource Discovery and Selection. The first task of the scheduler is resource discovery. The main goal is to identify a list of authorized hosts that are available to a given user. Most resource discovery algorithms interact with some kind of grid information service (GIS), like MDS (Monitoring and Discovery Service) in Globus [3]. Once the list of possible target hosts is known, the second phase of the broker is selecting those resources that are expected to meet the time or cost

[*] This research was supported by Ministerio de Ciencia y Tecnología through the research grant TIC 2002-00334.

constraints imposed by the user. In order to fulfill the user time restrictions the resource broker has to gather dynamic information about resource accessibility, system workload, network performance, etc. Moreover, if we contemplate an economic environment, the broker has to gather additional information about the price of the resources. Some environments like GRACE (Grid Architecture for Computational Economy) [4] provides a suite of trading protocols which enables resource consumers and providers to negotiate the cost of resources according to different criteria.

Job Scheduling. The next stage of resource brokering is job scheduling, i.e., the mapping of pending jobs to specific physical resources, trying to maximize some optimization criterion specified by the user. Most of the grid systems in the literature use performance-guided schedulers, since they try to find a job-to-resource mapping that minimizes the overall execution time (i.e. optimizes performance) [5] [6]. On the other hand, economy-guided schedulers include the cost of resources as optimization criterion [7] [8]. For example the Nimrod/G broker [7] allows users to specify a budget constraint (cost of resources), a deadline constraint (execution time), or both, and it incorporates different scheduling algorithms for cost optimization, and/or time optimization.

Job Migration. A grid is inherently a dynamic system where environmental conditions are subjected to unpredictable changes. In such a context, job migration is the only efficient way to guarantee that the submitted jobs are completed and the user restrictions are met. Most of the systems dealing with job migration face up to the problem from the point of view of performance [6] [9]. The main migration policies considered in these systems include, among others, performance slowdown, target system failure, job cancellation, detection of a better resource, etc. However, there are hardly a few works that manage job migration under economic conditions [8]. In this context, new job migration policies must be contemplated, like the discovery of a new cheaper resource, or variations in the resource prices during the job execution.

2 Resource Brokering in a Dynamic Economic Environment

In this work we investigate the influence of economic factors over dynamic resource brokering, in the context of the GridWay project [6]. We propose an extension of the MDS information service schema [3] in order to deal with economic information. Then, we evaluate different optimization criteria for job scheduling and migration, which combine both performance and cost information.

2.1 The GridWay Framework

The GridWay framework is a Globus compatible environment, which simplifies the user interfacing with the Grid, and provides the mechanisms for efficient execution of jobs on the Grid with dynamic adaptation to changing conditions. From the user point of view, the GridWay framework consists of two main components:

Command-Line User Interface. This interface significantly simplifies the user operation on the Grid by providing several user-friendly commands for submitting jobs to the Grid ("gwsubmit") along with their respective configuration files (job templates), stopping/resuming, killing or re-scheduling jobs ("gwkill"), and monitoring the state and the history of the jobs ("gwps" and "gwhistory"). For a given job, the template file must include the name of the executable file, its arguments, the name of the input and output files, the name of the restart files for checkpointing purposes in case of migration, and a per-job optimization criterion, which have to be maximized whenever is possible, when the job is scheduled on the Grid.

Personal Resource Broker. Each user interacts with its own personal resource broker, called Submission Agent. It is responsible for resource discovering, scheduling and submitting the user jobs, monitoring job performance, and migrating jobs when it is required. The scheduling policy is based on a greedy approach, so that the scheduler tries to maximize the optimization criterion specified by the user for each individual job, without considering the rest of pending, rescheduled or submitted applications. The migration of a job can be initiated by several events: (a) A rescheduling request sent by the user; (b) A failure in the target host; (c) A new better resource is discovered, which maximizes the optimization criterion selected for that job.

2.2 Extension of the MDS Schema and New Information Providers

To adapt the GridWay framework for dealing with economic information, it is necessary to extend the Globus MDS schema and design new information providers, which supply this kind of information to the Grid Resource Information Service (GRIS).

The extension of the MDS schema is achieved by adding to the LDAP directory a new structural object class called MdsEconomicInfo, and a new auxiliary object class called MdsCpuPrice. The attribute Mds-Cpu-Price-Per-Second of this object class contains the CPU price information generated by the resource provider, which uses an abstract monetary unit, called Grid Currency Unit (g.c.u.). In addition to these new object class and attributes, we have defined new auxiliary objects to manage the cost of other physical resources, like the cost of the memory space and disk space used by the program, or the cost of the network bandwidth consumed by the program. However, these elements are reserved for a future use, and they are not considered in this work.

The information provider for supplying the economic data is based on a simple implementation, since it reads the information about the cost of resources (CPU, memory, disk or network) from a file stored in each target host. In a future work, we plan to integrate the GridWay broker with the GRACE economic information providers [4] and GridBank [10], which include a suite of trading protocols for resource cost negotiation between resource consumers and resource providers.

3 Experimental Environment

3.1 The Benchmark: A Financial Application

The experimental benchmark used in this work is based on a financial application [11] [12] in particular, a Monte Carlo (MC) simulation for pricing European Call options. We briefly describe this problem.

Using the assumption of no arbitrage, the price of a derivate security can be computed as the expected value of its discounted payouts, where the expectation is taken with respect to the risk-neutral measure. In the particular case of a European call option, its price is the expected value of the payoff:

$$E\{e^{-r\Delta t} max(S(t+\Delta t)-X(t),0)\} \tag{1}$$

where t is the current time, r is the risk-free rate of interest, $X(t)$ is the exercise price, Δt is the holding period, and $S(t+\Delta t)$ is the stock price at time $t+\Delta t$.

Although Black and Scholes [13] provide an exact analytic method for pricing European options, numerical solutions are also very attractive, since they provide a general framework for solving this kind of problems, yet when an analytic model cannot be obtained. In particular, MC simulation exhibits significant advantages relative to other numerical models: it is a flexible technique, easy to implement, and inherently parallel, since random samples can be generated and evaluated independently. Furthermore, the error convergence rate in MC simulation is independent of the dimension of the problem, since the standard deviation of the MC estimation decreases at the order $O(1/\sqrt{N})$, where N is the number of simulations.

The MC approach for pricing options is based on simulating the changes in the values of the stock over the time horizon. The evolution of the asset, $S(t)$, can be modelled as a random walk following a Geometric Brownian Motion:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t) \tag{2}$$

where $dW(t)$ is a Wiener process, μ the instantaneous drift, and σ the volatility of the asset.

Assuming a lognormal distribution, using the It's Lemma, and integrating the previous expression over a finite time interval, δt, we can reach an approximated solution for estimating the price evolution of $S(t)$:

$$S(t+\delta t) = S(t)e^{(\mu-\sigma^2/2))\delta t+\sigma\eta\sqrt{\delta t}} \tag{3}$$

where η is a standard normal random variable.

To simulate an individual price path for a given holding period Δt, using a m-step simulation path, it is necessary to evaluate the price of the asset at each time interval: $S(t+\delta t)$, $S(t+2\delta t)$,..., $S(t+\Delta t)$=$S(t+m\delta t)$, $i=1,2,...,n$, where δt is the basic simulation time-step, i.e. $\delta t = \Delta t/m$.

To generate random numbers, we rely on the Scalable Parallel Random Number Generators (SPRNG) library, developed at the Florida State University [14].

This library includes different parallel random number generators, which can be used to develop a parameterized version of our Monte Carlo simulation algorithm. In our simulations, we have used the additive Fibonacci random number generator.

In particular, our experiment computes the expected price of a European Call Option over one year time horizon ($\Delta t = 1$), using a simulation time interval of one week ($\delta t = 1/52$), i.e., each simulation path is computed by a sequence of 52 time steps. The number of independent paths simulated is N=4 millions. The estimated price of the Call Option is given by the average value of the payoff computed for the 4 million simulations.

3.2 Results

In this section we investigate how the scheduling and migration decisions taken by the GridWay resource broker can change according to the optimization criterion specified by the user for a given job, and how these different decisions can affect to the overall execution time of the job, and the total CPU price.

Our Grid testbed consists of three Sun Workstations with Solaris 8, whose main characteristics are summarized in Table 1.

Table 1. Characteristics of the machines in the research testbed.

host	Model	Speed	Memory	Perform.(peak)
sunblade	Sun Blade 100	500MHz	256MB	1000 MFLOPS
ultra1	Sun Ultra 1	167MHz	128MB	334 MFLOPS
sun250	Sun Enterprise 250	296MHz	256MB	600 MFLOPS

In the subsequent experiments, we assume that the cost of different resources exhibits the following behavior (see Figure 1): When the program execution starts, all the three hosts on the grid charge the same CPU price per second (12 g.c.u.). Around two minutes later, the sun250 and ultra1 hosts reduce the CPU price to 6 g.c.u., and 5 g.c.u. per second respectively.

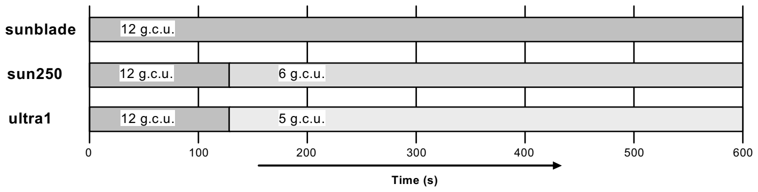

Fig. 1. Dynamic changes in the CPU prices of the testbed hosts.

Next, we analyze the different schedules generated by the resource broker under three different optimization criteria specified by the user:

Criterion #1: Performance * CPU-Free(%). This optimization criterion tries to minimize the overall execution time of the application. If the user establishes this criterion for the job, and assuming that all the machines are idle, the resource broker allocates the job to the host with highest performance, i.e., the sunblade host. Since changes in CPU prices have no effect on performance, the scheduler does not trigger any migration and the program is executed entirely on the initial host. Table 2 (#1) shows the time and the price of this schedule. Notice that the overall elapsed time includes de user and system CPU times, the I/O time, and also the overheads due to the transmission of the executable file, and the input and ouput files.

Criterion #2: 1 / CPU-Price-per-Second. This optimization criterion tries to minimize the total amount that the user pays for the CPU usage. To compute this amount only the user CPU time expended by the job is considered. Other times, like the system CPU time, the I/O time, or the CPU time expended by other processes are not considered. Using this optimization criterion, the resource broker will submit the job to the cheapest resource on the Grid. If two hosts are the same price, the broker selects that with maximum performance. Under this optimization criterion, the program execution starts on the sunblade host, which exhibits the highest performance at the same price. Two minutes later, when sun250 and ultra1 CPU prices change, the resource broker migrates the job to the new cheapest host, i.e., the ultra1 host. Table 2 (#2) displays the results for this schedule. As we can observe, this optimization criterion improves neither the price nor the time with respect to the first schedule. This is due to the low performance of the ultra1 host, which takes a long time to execute the program. Consequently, the total accumulated CPU price is higher than the previous case, despite the lower price per second of the ultra1 host.

Criterion #3: Performance / CPU-Price-per-Second. To avoid worthless migrations, we consider this third optimization criterion, which tries to minimize the performance to CPU price ratio. If the user specifies this criterion for the job, the program execution starts on the sunblade host, which exhibits the best trade-off between performance and price (see Table 3). When sun250 and ultra1 prices change, the resource broker migrates the job to the sun250 host, which maximizes now the optimization criterion (see Table 3). The results displayed in Table 2 (#3) show that this schedule gets an 8.2% reduction in the total CPU price with respect to the criterion #1, against a 32.4% increment in the overall elapsed time.

Figure 2 compares graphically the accumulated price and the overall elapsed time of the three schedules.

Table 2. Results for different schedules.

Criterion	CPU time (s)	Total Price (g.c.u.)	Elapsed time (s)
#1 Perf. * CPU-Free(%)	218.6	2,623.6	271.2
#2 1/CPU-price	103.2 + 348.6	2,981.7	553.3
#3 Perf./CPU-price	103.2 + 195.1	2,409.4	359.2

Table 3. Performance to CPU-price-per-second ratio for the testbed hosts.

Host	CPU-Price	(Perf./CPU-price)	CPU-Price	(Perf./CPU-price)
sunblade	12 g.c.u.	83.3	12 g.c.u.	83.3
sun250	12 g.c.u.	50.0	6 g.c.u.	100.0
ultra1	12 g.c.u.	27.8	5 g.c.u.	66.8

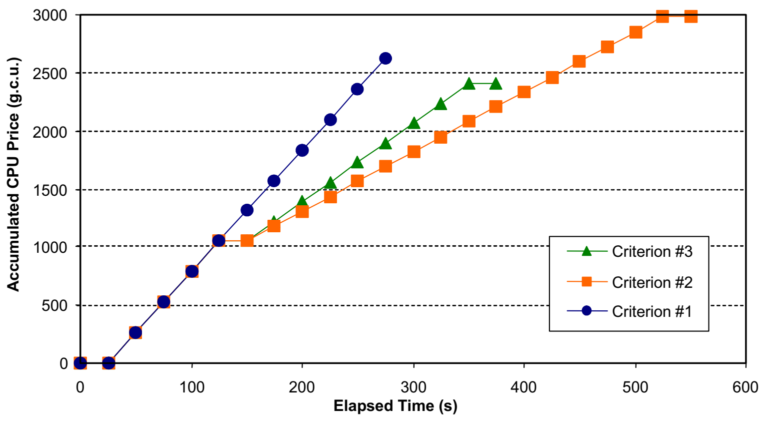

Fig. 2. Comparison of the schedules with different optimization criteria.

4 Conclusions and Future Work

This paper is focused on the problem of dynamic resource brokering in the presence of economic information. An extension of the MDS information service schema, in the context of the GridWay project has been proposed, in order to deal with this new kind of information. We have evaluated several optimization criteria for job scheduling and migration, and we conclude that, in order to reduce the overall CPU cost, it is important to use optimization criteria based on both performance and CPU price measures.

In a future work, we plan to incorporate to the brokering model the cost of other physical resources, like the cost of the memory and disk space used by the program, the cost of the network bandwidth consumed, etc. Other improvements contemplated include the evaluation of alternative optimization criteria, the development of new information providers, and the integration with the GRACE and GridBank environments, which supply a complete suite of economic information providers and trading protocols.

References

1. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1998)
2. Schopf, J.: A General Architecture for Scheduling on the Grid. Special issue of JPDC on Grid Computing (2002)
3. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. 10th IEEE Int. Symp. on High-Performance Distributed Computing (2001)
4. Buyya, R., Abramson, D., Giddy, J.: An Economy Driven Resource Management Architecture for Global Computational Power Grids. Int. Conf. on Parallel and Distributed Processing Techniques and Applications (2000)
5. Raman, R., Livny, M., Solomon, M.: Matchmaking: Distributed resource management for high throughput computing. Int. Symp. on High Performance Distributed Computing (1998)
6. Huedo, E., Montero, R.S., Llorente, I.M.: An Experimental Framework For Executing Applications in Dynamic Grid Environments. NASA-ICASE Technical Report 2002-43 (2002)
7. Abramson, D., Buyya, R., Giddy, J.: A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. Future Generation Computer Systems Journal, Volume 18, Issue 8, Elsevier Science (2002) 1061-1074
8. Sample, N., Keyani, P., Wiederhold, G.: Scheduling Under Uncertainty: Planning for the Ubiquitous Grid. Int. Conf. on Coordination Models and Languages (2002)
9. Allen, G., Angulo, D., Foster, I., and others: The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment. Journal of High-Performance Computing Applications, Volume 15, no. 4 (2001)
10. Barmouta, A. and Buyya, R., GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration. 17th Annual Int. Parallel and Distributed Processing Symposium (IPDPS 2003), Workshop on Internet Computing and E-Commerce (2003)
11. Moreno-Vozmediano, R., Alonso-Conde, A.B.: A High Throughput Solution for Portfolio VaR Simulation. 4th WSEAS Int. Conf. on Mathematics and Computers in Business and Economics (2003) in press.
12. Branson, K., Buyya, R., Moreno-Vozmediano, R., and others: Global Data-Intensive Grid Collaboration. Supercomputing Conf. (SC2003), HPC Challenge Awards (2003)
13. Dupire, B.: Monte Carlo Methodologies and Applications for Pricing and Risk Management. Risk Books, 1st edition (1998)
14. Mascagni, M., Srinivasan, A.: Algorithm 806: SPRNG: a scalable library for pseudorandom number generation. ACM Trans. on Mathematical Software (TOMS), Vol. 26, Issue 3, September (2000) 436-461