Dipartimento di Informatica
Università del Piemonte Orientale "A. Avogadro"
Spalto Marengo 33, 15100 Alessandria
http://www.di.unipmn.it

TECHNICAL REPORT TR-INF-2004-01-01-UNIPMN
(January 2004)

Grid scheduling and Economic Models

Author: Massimo Canonico (canonico@mfn.unipmn.it)

Abstract

Computational Grids are becoming attractive and promising platforms for solving large-scale (problem solving) applications of multi-institutional interest. However, the management of resources and scheduling computations in the Grid environment is a complex undertaking as they are (geographically) distributed, heterogeneous in nature, owned by different individuals or organizations with their own policies, different access and cost models, and have dynamically varying loads and availability. This introduces a number of challenging issues such as site autonomy, heterogeneous substrate, policy extensibility, resource allocation or co-allocation, online control, scalability, transparency, and economy of computations . Resource allocation complexity due to decentralization and heterogeneity is also present in human economies. In general, modern economies allocate resources in systems whose complexity overwhelms any algorithm or technique developed for computer systems. In this paper, we describe and comment different approaches present in scientific literature that use economic models to study the resource allocation problem in the Grid environment.

1 Introduction

As described in [1], more applications are turning to Grid computing 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 well scheduled. *Grid Scheduling* is defined as the process of making scheduling decisions involving resources over multiple administrative domains. This process can include searching multiple administrative domains to use a single machine or scheduling a single job to use multiple resources at a single site or multiple sites. We define a job to be anything that needs a resource and we use *resource* to mean anything that can be scheduled: a machine, disk space, a QoS network, and so forth.

In managing such complex environment, traditional approaches to resources management that attempt to optimize system-wide measure of performance cannot be employed. Traditional approaches use centralized policies that need complete state information and a common fabric management policy, or decentralized consensus-based policy. Besides, these approaches do not take into account the different scheduling and management preferences of users and resource owners. As discussed in [2], many projects propose and explore the usage of economic-based paradigms for managing resource allocation in Grid computing environments. The economic approach provided a fair basis in successfully managing decentralization and heterogeneity that is present in human economies. Competitive economic model provide algorithms/policies and tools for resource sharing or allocation in Grid systems. The models can be based on bartering or prices. In the *bartering-based model*, all participants need to own resources and trade resources by exchanges (i.e., storage space for CPU time). In the *price-based model*, resources have a price, based on the demand, supply, value and the wealth in the economic system.

In this paper, we illustrate and comment different approaches to make scheduling on Grid environment using economic models presented in scientific literature. In section 2, we fix the objectives and in the section 3 we present the different approaches. Finally, in section 4 discuss the state of art.

2 Motivations and goals

In designing resource allocation and control mechanisms in Grid environment several goals need to be considered. Some of the important goals are outlined below, while in later sections, we present

different approaches that try

- to decentralized resource access, allocation and control mechanisms;
- to design reliable, fault-tolerance and robust allocation mechanisms;
- to provide guarantees to users and applications on performance criteria. Some of the performance criteria in distributed systems include the following:
  - response time
  - throughput
  - application failure probability
  - network QoS
- to provide a unified framework in which users have transparent access to the services of a distributed system, and services are provided in an efficient manner. This framework should hide complexity of multiple resource suppliers and resource allocation policies.

Some of these goals could be realized using economic models that provide several interesting contributions to resource sharing algorithms. The first is a set of tools for limiting the complexity by decentralizing the control of resources. The second is a set of mathematical models that can yield several new insights into resource sharing problems.

In an economy, decentralization is provided by the fact that economic models consist of agents which selfishly attempt to achieve their goals. There are two types of agents, *suppliers* and *consumers*. A consumer attempts to optimize its individual performance criteria by obtaining the resources it requires, and is not concerned with system-wide performance. A supplier allocates its individual resources to consumers. A supplier's sole goal is to optimize its individual satisfaction (profit) derived from its choice of resource allocations to consumers.

Most economic models introduce money and pricing as the technique for coordinating the selfish behavior of agents. Each consumer is endowed with money that it used to purchase required resources. Each producer owns a set of resources, and charges consumers for their use. The price a producer charges for a resource could be determined by its supply and the demand of the agents for resource. Several papers apply versions of this model to decentralized resource allocation in computer systems.

3 Approaches

In this section, we present and comment different ways to combine economic models and computational Grid. In particular we are interested about three fundamentals aspects: (i) the *scheduling policy* that specifies how to assign the jobs to resources (ii) the *utility function* to quantify the goodness of each assignment and (iii) the *market model* used and how the *prices* vary for each resource.

Economic models for resource management and scheduling in Grid computing

The Nimrod/G resource broker [3] is a global resource management and scheduling system (see Figure 1) that supports deadline and economy-based computations in Grid computing environments

for parameter sweep applications. Parameter studies involve the execution of a large number of (independent) tasks over a range of parameters. Scheduling of such applications appears simple, but complexity arises when users place SOS constraints like execution time and cost limitations. Such a guarantee of service is hard to provide in a Grid environment as its resources are shared, heterogeneous, distributed in nature, and owned by different organizations having their policies and charging load and at the same time meet cost constraints. In the Nimrod/G application level resource broker (also called an application level scheduler) for the Grid, three adaptive algorithms for scheduling are incorporated:

- time minimization, within time and budget constraints;
- cost minimization, within time and budget constraints;
- none minimization, within time and budget constraints.

The Time Minimization algorithm attempts to complete an experiment as quickly as possible, within the available budget. The Cost Minimization algorithm attempts to complete an experiment with the lowest cost as possible within the deadline. A final Algorithm ("None Minimization") attempts to complete the experiment within the deadline and cost constraints without minimizing either.

Therefore, in this model, the utility functions are only two:

- *time(R,J)* that returns the execution time of job J in the resource R;
- *cost(R,J)* that returns the cost of the execution of job J in the resource R.

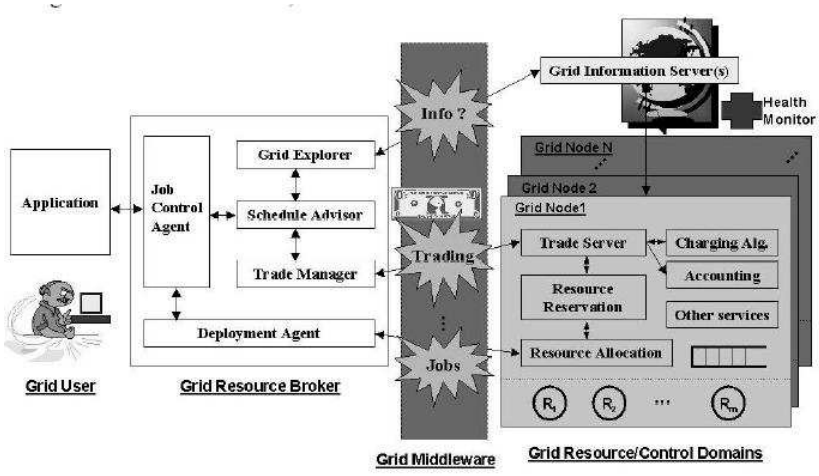

Figure 1: Nimrod/G resource broker

To evaluate the different algorithms for scheduling, two scheduling experiments have been carried out for a given deadline of 4 hours (with 18 minutes of extension for the second experiment) and a budget of 250 000 (G$ or tokens) with different optimization strategies (time minimization and cost minimization).

In these scheduling experiments, the Nimrod-G resource broker employed the *commodity market* model to establish a service access price. The access price varies from one consumer to another and from time to time, as definite by the resource owners. Depending on the deadline and the specified

budget, the broker develops a plan for assigning jobs resources. The access price has been established dynamically using GRACE (GRid Architecture for Computational Economy) [3] resource trading protocols (commodity market model), but is based on an arbitrary assignment for demonstration purposes only.

The resources are characterized only by the number of CPUs, this parameter is used to evaluate the computational time of the jobs (the computation time is considered a parameter easy to calculate and set to a constant value). In the experiments described, the resource price has been assigned arbitrarily and it's constant (it doesn't respect the commodity market model). Besides, the results have not been compared with any classical scheduling approach (i.e., Round-Robin), thus it's difficult evaluate the results obtained. Finally, the algorithms used to calculate the utility functions could require long time to be completed. For example, in time minimization the first step consists on calculating the next completion time for an assigned job for all resources. In Grid environment the number of the resources could be high, and calculate the computational time of a job for each possible resources could require long time.

Market-Based Proportional Resource Sharing for Clusters

Also the researchers of the University of California at Berkeley propose a market-based approach [4] to cluster resource management based on the notion of a computational economy which optimizes for *user value*. The architecture of the system provides a proportional-share schedulers as the resource managers for all basic computational resources (CPU, memory, network, I/O). The reasons are: (i) proportional-share schedulers provide an intuitive model of resource allocation, (ii) there exist efficient algorithms for implementing them, and (iii) they provide flexibility in exposing differ entities to which to assign value (i.e., in addition to assigning value simply based on shares, given a CPU stride schedulers, reasonable user estimate of CPU time needed, and admission control one could potentially build a market-based system based on deadlines). In this model, resource rights for a shared resource are encapsulated as tickets. A resource is represented by a total of T tickets. An application holding t tickets competing for use of that resource obtain a share of t/T of the resource. For example, with CPU stride scheduling, an application holding t tickets would obtain an accurate t/T allocation of the CPU over time. The model used is a intuitive policy based on opportunity-cost charging; in this scheme, n jobs are competing for some shared resource r (in this case CPUs on node). Job i has a stated value b_i, which may or may not be the same as true value v_i (i.e., suppose a user is trying to undervalue the value of an application to save credits or perhaps the user does not realize how important a particular run is). The system allocates shares and charges as follows. If $n = 1$, there is only one user demanding the resources. Thus, that user is imposing no burden on the system and not denying anyone else an opportunity since there is no competition for r. In this case, that user is charged nothing and is given all for r. If $n > 1$, multiple jobs are competing for r and jobs receive shares as they would if the b_i's were tickets. That is, job i with stated value b_i obtains $b_i / \sum_{j=1}^{n} b_j$ of r over time. When $n > 1$, job i is charged b_i credits per minute. If multiple jobs are competing, each with $b_i = 0$, each job receives fair share and is not charged. The computing of shares and charging is computed dynamically as jobs join and leave system. Another nice property of this policy is that it is computationally simple to implement. The overhead required to compute allocations and charging are straightforward and are even simpler than those needed to implement stride scheduling. Furthermore, because we do not anticipate users using the cluster to run large numbers of very short jobs (i.e., less than a second), the dynamic updating of allocations, which require system calls, should incur fairly minimal overhead.

The main contribution of this article is the discussion about the systems that implements market-based ideas to cluster resource management. The models studied (Ferguson, Spawn, Popcorn, Mariposa, SC centers) applies different interpretation of tokens distribution and what resources share, but the main observation regards that most of the work so far has focused mainly on the economic front-end and layer. Very little attention has been paid to the end-to-end problem: how real users and applications use marked-based systems in practice, and implementations real usage has on other layers of the system (i.e., Popcorn requires each program to be written with the economy in mind using a special API). Besides, Spawn and Popcorn assume dedicated use of the system. Most of all systems considered use only CPU speed to make decision in scheduling process. They don't consider Usually the resource to other parameter (i.e., disk space, network bandwidth, etc.).

Marked-based Resource Allocation for Grid Computing: A Model and Simulation

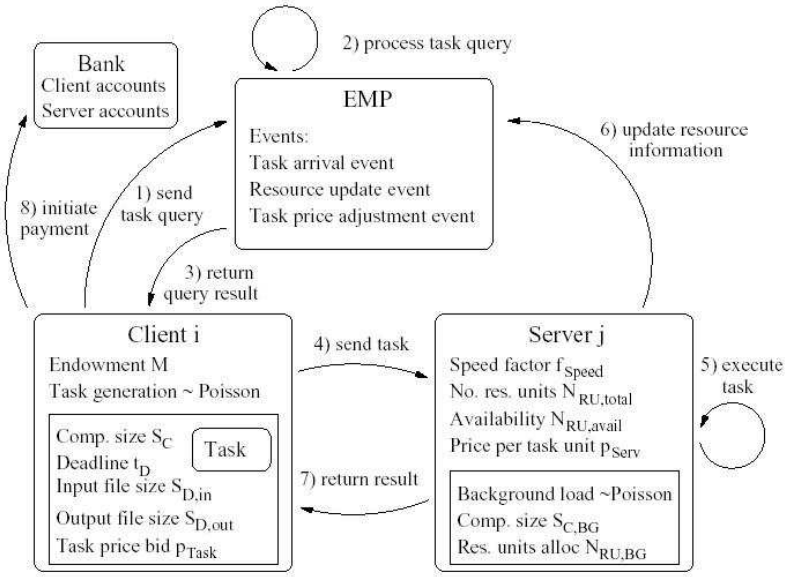

Figure 2: Model of the marketplace

This model represents an electronic marketplace for distributed computational resources [5]. As shown in Figure 2, there are three main actors in the model, which are assumed to be over Internet: the Clients, the Servers and the Electronic Marketplace (EMP). Clients generate tasks which required computational resources for their execution. The Servers provide these resources: they advertise and sell them at the EMP. More in detail:

- a task which is created by a Client is characterized by the size of its computation S_C (in task units), the size of the input and output files, its price bid and a deadline;
- each Server has a resource consisting of one or more resource units (RU). This allows to model either time-shared resources (where a resource unit corresponds to a time share of a resource)

or space-shared resources (where a unit corresponds to a CPU).

About *scheduling policy*, there are different ways of scheduling resource units to tasks which have been allocated by the EMP. In this model has been considered the following two cases:

- allocate all available resource units of the Server to the task;
- allocate a fraction of the currently available resource which is proportional to the task's price bid $p_{Task,i}$, in relation to the sum of all price bids $\sum p_{Task,i}$ on the Server (including the bid of the task itself).

The EMP provides facilities for the Servers to advertise their resource. The parameters to be published include the number of resource units, the price per task unit and the resource's speed. For the Clients (or their agents) it provides means to search for a suitable resource and negotiate the price. Currently, the processing delays of the Clients and the Servers requests are neglected (until results from 'real-world' experiments are available).

Three different resource allocation protocols are discussed: the Round-Robin Protocols (RR), Continuous Double Auction Protocol (CDA), and Proportional Share Protocol(PSP). In the RR no pricing is used. The incoming task queries are matched with the 'next' available resource offer which meets the task's constraints but which is usually not the 'best'. For this purpose an iterator is used which cycles through the list of Server resource offers.

The aim of CDA, instead, is to allocate the best possible resource to an arriving task and to priorities tasks according to their price bid. Finally, in the PSP every time a task or background task (the tasks which are outside the control of the EMP, i.e. the private tasks of the resource owner) starts or completes execution on the Server, the other tasks need to be rescheduled. The tasks' resource shares change, and so do their execution speed (the effective execution speed depends on the load at the Server).

It is often assumed that market-mechanisms are better than conventional round-robin approach. Exploring different scenarios, it's possible to assert that, for a cluster of homogeneous resources, the CDA will perform best. However, if the load is low, the differences between the three protocol are small, and using the computationally less expensive RR might be sufficient. For a situation where there is a choice of resources with different quality or load, as it is the case in a computational Grid, the results of RR will be worse for the two market-based protocols. This is due to their allocation of the fastest possible resources and the prioritizing of tasks with high weights. The CDA will perform best in most cases. However, for a high number of resources the performance of the PSP will be comparable.

Also in this case a fundamental parameter for scheduling decision is the computation time for each job. Correctly for a distribute environment, the model consider the link bandwidth of the resource as a parameter to use during scheduling process. The reschedule operation (stop the job and assign it to another resource) is considered a priceless operation (in time and money). In this models, the resources are characterized by CPU speed, bandwidth link and price (no disk space, physical memory or reliability).

Bootstrapping Distributed Computational Economy with Peer-to-Peer Bartering

In this article [6], it presents a baseline architecture for bootstrapping economies based on *peer-to-peer bartering*, with an eye to its support in the PlanetLab network testbed [8]. The architecture

consists of three pieces: (i) resource discovery, (ii) secure resource peering, and (iii) bartering. For our interest, we describe only the bartering piece.

Bartering strategies specify how peers negotiate exchange rates for peering and how peers execute the peering protocol. Negotiating exchange rates involves determining what amount of resources a peer X exchanges with a peer Y as part of the peering and how many such exchanges will occur.

One simple strategy based on reciprocity that has proven to be remarkably robust and effective against a wide range of competing strategies is TIT FOR TAT [7]. TIT FOR TAT is the strategy of beginning with cooperation and, thereafter, doing whatever the other peer did in the previous round. It is simple, encourages cooperation, punishes defection (but is forgiving), and in practice outperforms virtually all competing strategies in a number of situations. Given this, one natural strategy for bootstrapping a computational economy is to start with P2P TIT FOR TAT, where resource exchange in a round is rewarded with resource exchange in the next round and reneging in a given round is punished by reneging in the next round.

In environments where the set of peers is fairly static and peers tend to interact with large numbers of other peers, P2P TIT FOR TAT is an appropriate strategy. In environments where the set of peers is large and dynamic, the probability that any two peers interacts decreases. In such environments, more sophisticated strategies will need to be employed.

The article does not present simulation part to evaluate the bartering model.

Grid environment is quite unpredictable, and if the bartering works well in a high cooperative environment we can not do this assumption for Grid. This could be a ulterior aspect to drive the further studies the using of economic model with price.

Economic Models for Allocation Resources in Computer Systems

This paper [9] presents an economic approach price-based to resource allocation in computer systems using the case studies of load balancing and data management in systems, flow control, QoS provisioning in management in Integrated Services Networks, and Multiple Access Protocols in Broadcast Packet Networks. For our purpose, we are interested only in the *Load Balancing Economy*. In Load Balancing Economy there are N processor connected via point-to-point network. A link e_{ij} is a connection between P_i and P_j. Each link has a delay variable which is d_{ij}. The service rate of each processor P_i is r_i. The resources are processor time (CPU time) and the communication bandwidth. Jobs enter the distributed system and request resources based on the prices. Job j has a service time of μ_i, and it purchases μ_j/r_k time units on P_k. Jobs are various preferences on the services they wish receive. The preferences are the following:

- Price Preference(PP): Jobs prefer service to be done as cheaply as possible, the cost C_k is composed of the cost of accessing service at processor at processor P_k.

$$\min_{\forall k}[C_k]$$

- Service Time preference(ST): Jobs prefer the element of the budget set which gives the least response time. Job j located on P_i computes the service time at processor P_k from the following:

$$\min_{\forall k}[ST_k = \mu_j/r_k + InputByte * d_{ik} + OutputByte * d_{ki}]$$

  where $InputByte$ and $OutputByte$ are respectively the size in bytes of input and output data.

- Service and price preference (SVTP): jobs place a relative preferences of service time over cost at a processor:

$$\min_{\forall k}[C_k + A * ST_k]$$

where A is a weight giving relative importance to ST.

The paper considers four types of auctions considered by this economy. The first is an *English auction* where the price of the resource is gradually increase with the bidding. The highest bidder obtains the resource. The second is the *Dutch auction*, where prices are gradually decreased if no bid is submitted. The third is the *Hybrid auction*, where the asking price of a resource is increased if a bid is submitted, and decrease if no bid is submitted. The fourth is the *Sealed Bid auction*, where sealed bids are submitted by the agents and the highest bid is given access to the resource. In this auction model, agents are not aware of the amounts bid by the other consumers. Jobs (when they arrive) perform three operations to purchase resources which are: compute the budget set, find the most preferred elements (demand set) of the budget set, and generate a bid for demand set. The processors auction resources (link and CPU time) to the jobs, advertise resource prices in local bulletin boards and neighboring processors, and update prices based on auctioning results and arriving price updates from neighboring processors.

The load balancing problem is to design algorithms that minimize the mean job waiting time by migrating the jobs to balance the workloads of the processor nodes. Each job independently computes the best place (node) to be served based on its preferences and wealth, and the resource prices. The main goal of each processor (node) in the economy is to maximize revenue.

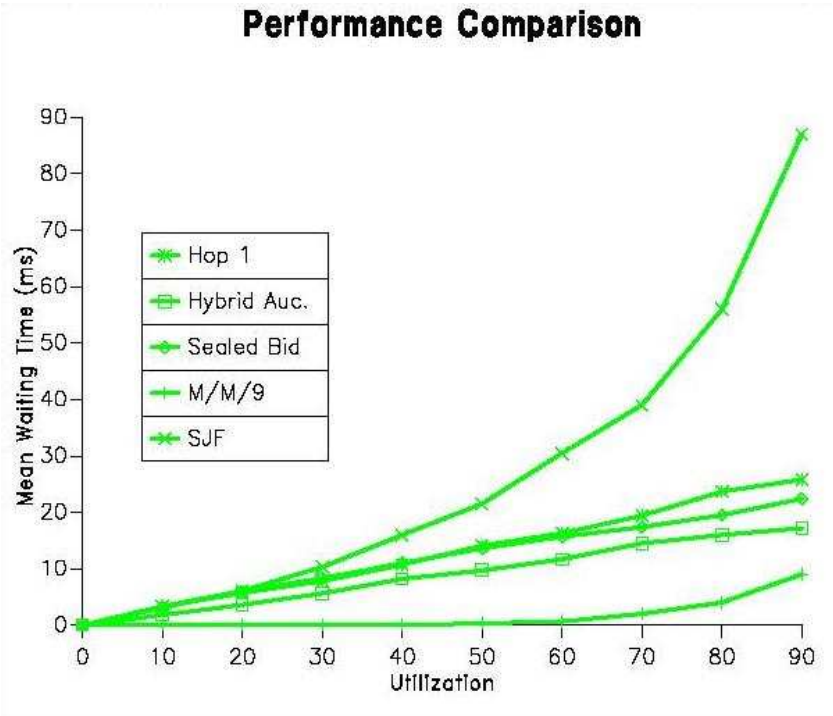

Figure 3: Market economy scheduling architecture

Figure 3 plots the mean job waiting times as a function of the system utilization. The figure shows that Hybrid and Sealed-Bid actioning based economies perform better than the HOP 1 algo-

rithm which is a non economic algorithm, and the Hybrid-auction economy performed the best at all utilizations. At low utilizations the Sealed-Bid economy is as good as HOP 1, and better at high utilizations. The SJF (shortest job first on an $M/M/1$ system) algorithm performed the worst.The $M/M/9$ case is depicted for comparison purposes only. The $M/M/9$ is a queuing system where 9 processors serve one queue, and there is no communication delay, and all the informations is globally available and exact.

An interesting phenomena in the load balancing economy is that jobs migrate in search of suppliers based on the job preference model. Numerical studies on this phenomena indicated that jobs with response time preference migrated less compared to the jobs that have price preference. Jobs which had a combination of service time and price preferences migrated based on the weights given to service time or price.

This case study shows that for the load balancing problem, competitive economic concepts can achieve better levels of performance when compared to non-economic algorithms. In Ferguson's model, multiple sequential jobs compete for dedicated slices of CPU time on a collection of heterogeneous (only in speed) machines interconnected by point-to-point links. With this heterogeneity, jobs may be submitted on any machine, migrate to different machines for execution. In Grid Environment the machine are highly heterogeneous (not only in speed), a job could not be migrate/execute in any resources available. Also in this case, the utility function is used only for cost minimization or computational time minimization.

The models presented use concepts of mathematical economics were used to develop effective market based control mechanisms. However, there are drawbacks to this form of modeling where several agents have to use market mechanisms to decide where to obtain the service (which supplier?). If the demand for a resource varies substantially over short period of time, then the actual prices of resources will also vary causing several side effects such as indefinite *migration* of jobs between suppliers, and price guarantees over shorter periods of time.

Applying Economic Scheduling Methods to Grid Environments

In comparison with other economic systems, in paper [10], the model uses individual utility functions for the different Grid participants. Resources in a domain are locally controlled by a $MetaManager$ that acts as a broker or trade with MetaManagers from other domains (see figure 4). Each domain can act independently and may have different objective policies. Also, each job request can include an individual objective function. This function can be defined using a description language that is then evaluated to scalar value at run time. The scheduling system combines the different objective functions to find the *equilibrium* between supply and demand (more information about equilibrium can be find at [11]).

All users submit their jobs to their local MetaManager. This MetaManager is responsible for the whole equilibrium process. During job submission, the user may specify requirements for the job execution (i.e., estimate run time, earliest start time, and a latest completion time). Most current job scheduling systems already require the specification of a maximum run time by the user, since this information is required for some scheduling strategies, for example backfilling. If the job finishes earlier than estimated, the idle resources can be used by other submitted jobs.

After the local MetaManager has received a new job, the local domain scheduler first analyze the requirements of the job. Next, local offers are generated if the local resources match the job specifications. Only the best local offers are kept for further processing. The job is forwarded to other connected domains. To prevent the system for permanently forwarding job request, a user can restrict the number of hops made or specify a time-to-live for the request forwarding. The remote

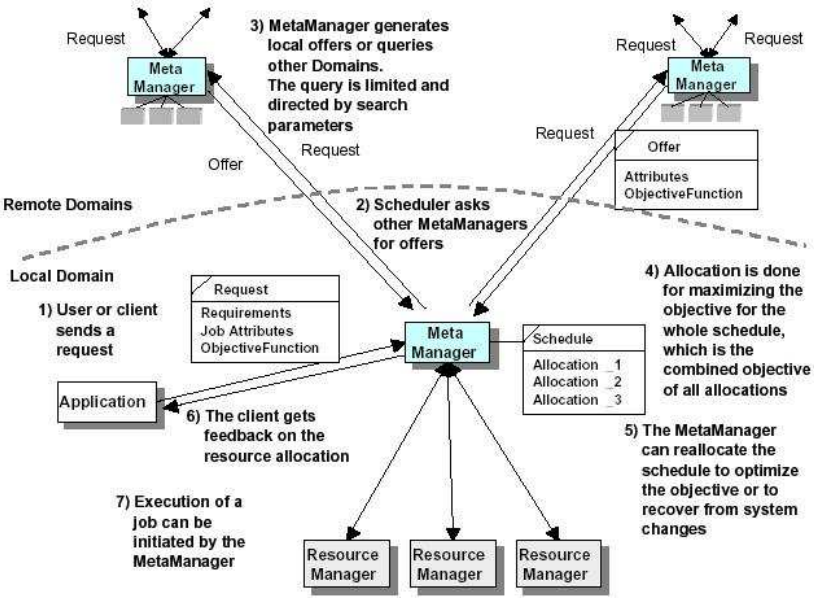

Figure 4: Market economy scheduling architecture

domains create new offers and send their best offers back to the original domain. Finally, the best of all incoming and all locally generated offers is selected as the final offer.

A basic component of an economically driven approach is a description language that is used to specify requests, resources and offers. An important feature of this language is the ability to describe resources, request, and offers, as well as individual objective functions, with the same basic statements. The description language is not limited to a certain resource type, so new resource classes can be added easily. The list of parameters that are available for the request formulation is presented in figure 5.

Figure 6 illustrates a request formulation. This example request includes assignments for a set of keys (*Hops, MaxOfferNumber,..,JobBudget*) and a utility function. The utility value depends on two conditions: the operating system and the number of available processors. The system try to maximize the $UtilityValue$, which for our example means that the job should be started as soon as possible and that the job costs are minimized.

The results of the simulations for different workload sets, different resource, configurations, and several different parameter settings for the objective functions show that the economic scheduling system is competitive with the conventional scheduling systems in term of the average weighted response time.

The economic approach provides several additional advantages over conventional approaches, such as allowing for site autonomy and the ability to use heterogeneous resources. The utility/objective functions for each Grid user and resource owner can be defined separately for each job request.

The most important aspect of this model regards the presence of MetaManagers: this is the first paper that proposes a scalable and modular solution about the scheduling process. In this model, the user could specify the computational time of job but also the *start time* (within the job should be start to execute) and the *deadline* (the time within the job must be finished). These parameters are useful to manage the backfilling of processes on resources, but they could not be easy to specify. Besides, the model does not describe how manage the case which the job is still running after the

Parameter	Description
Hops	This attribute limits the query depth to remote domains.
RequestID	This is a unique number that denotes the current request.
MaxOfferNumber	This is the maximum number of offers a user wants to receive. A value of 0 specifies that the MetaManager should automatically select the best offer according to the UtilityValue.
OfferBudget	This specifies the budget that is available for offer generation.
ReservationTime	This is the time until which the available resources should be reserved.
StartTime	A job must not be started before this date.
EndTime	A job must not end after this date.
SearchTime	The scheduling system can search for offers until this time instance.
JobBudget	This parameter specifies the maximum execution cost.
ReservationBudget	This parameter specifies the maximum reservation cost.
RunTime	This parameter specifies the execution time of a job.
UserName	This parameter specifies uniquely the submitting user.
Memory	This is the memory requirement per processor (in kBytes).
NumberOfProcessors	This is the number of requested resources.
UtilityValue	This value denotes the marginal gain from the user's point of view.

Figure 5: Simulation Topology

```
REQUEST "Req001" {
  KEY "Hops" {VALUE "HOPS" {2}}
  KEY "MaxOfferNumber" {VALUE "MaxOfferNumber" {5}}
  KEY "StartTime" {VALUE "StartTime" {900000}}
  KEY "EndTime" {VALUE "EndTime" {900028}}
  KEY "SearchTime" {VALUE "SearchTime" {899956}}
  KEY "JobBudget" {VALUE "JobBudget" {900.89}}
  KEY "Utility" {
     ELEMENT 1 {
       CONDITION{ (OperatingSystem EQ "Linux")
         && ((NumberOfProcessors >= 8)
         && (NumberOfProcessors <= 32))}
       VALUE "UtilityValue" {-StartTime}
       VALUE "RunTime" {43*NumberOfProcessors}
     }
     ELEMENT 2 {
       CONDITION{(OperatingSystem EQ "AIX")
         && ((NumberOfProcessors >= 8)
         && (NumberOfProcessors <= 64))}
       VALUE "UtilityValue" {-JobCost}
       VALUE "RunTime" {86*NumberOfProcessors}
     }
  }
}
```

Figure 6: Simulation Topology

deadline specified. Also in this model, the prices of the resources and how they vary to reach the equilibrium is not clearly specified. The resource are characterized by CPU speed, and memory space.

4 Discussion

In this paper, we have described some models published in scientific literature to manage scheduling in the Grid environment using economics models. The first consideration regards the characterization of the resources, the aspect upon which depends the complexity of the models. Usually, we have observed that the CPU speed is the main characteristic considered to make scheduling decisions and only fews models consider other parameters (i.e., memory (speed and space), link bandwidth or reliability). Using fews characteristics to describe the resources, consequently the demand can't be detailed enough to satisfy the user requirements. For instance, a user should formulate his requirements considering different hardware characteristics (CPU speed, bandwidth, reliability, memory) with different weights for each characteristic. A user request with high bandwidth weight means that the user consider it very important for his job. The combination of each weight with the hardware characteristic could be a useful utility function. In the models proposed, the utility functions regard only computational time and computational cost. In this survey, we can observe that the execution time of job on a resource seems to be a indispensable parameter for the scheduling process. To calculate it, usually the model propose to associate at each resource a "service rate" parameter and at each job a "service time" parameter. The first indicates how much fast the resource could execute a job and the second indicates how long is the computational time of the job. The quotation of this two parameter indicates the time units of the job. There are different articles that describe how it is difficult to specify this parameter [12].

Another parameter difficult to calculate is the price of the resources. Many papers cite the "commodity market model" as the solution of the problem to evaluate the cost of each resource. This model indicates that the price depends on supplies and demands, but it does not provide a specific algorithm to calculate the price.

Some papers consider the Computational Grid as a "large-cluster" where each node are quite similar (not heterogeneous resources), the links between nodes are fast and reliable (migration of processes from node to another node is easy and quick) and scheduling decision process could be execute to only one resource (as the front-end of a cluster). Unfortunately, Grid could not be compared to a "large-cluster" because the resources could be highly heterogeneous, distributed around the world and could be increase to large number. Only the last paper has proposed a scalable and modular solution for the scheduling process using a hierarchical organization of MetaManagers.

Reliability of the resources is one important characteristic in any environment where it's possible specify the QoS. In the papers read, there isn't any reference to this problem.

References

[1] J.M. Schopf *Ten actions when grid scheduling*, Chapter 2 in Grid Resource Management for Grid Computing, 2003.

[2] R.Buyya, D.Abramson, J.Giddy and H.Stockinger *Economic models for resource management and scheduling in Grid computing.* The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, May 2002.

[3] R.Buyya, D.Abramson and J.Giddy *An Evaluation of Economic-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications.* In 2nd International Workshop on Active Middleware Services (AMS '00), Aug. 2000.

[4] B.Chun and D.Culler *Market-Based Proportional Resource Sharing for Clusters.* Technical Report CSD1092, University of California at Berkeley, January 2000.

[5] J.Gomoluch and M.Schroeder *Market-based Resource Allocation for Grid Computing: A Model and Simulation.* 1st International Workshop on Middleware for Grid Computing (MGC2003), Rio de Janeiro, Brazil, June 2003.

[6] B.Chun, Y.Fu, A.Vahdat *Bootstrapping a Distributed Computational Economy with Peer-to-Peer Bartering.* Workshop on Economics of Peer-to-Peer Systems, 2003.

[7] R.Axelrod *The Evolution of Cooperation*, Basic Books, 1984.

[8] *www.planet-lab.org.*

[9] D.F.Ferguson, C.Nikolaou, J.Sairamesh, Y.Yemini *Economic Models for Allocating Resources in Computer Systems.* World Scientific Press, 1996.

[10] C.Ernemann and R.Yahyapour *Applyin Economic Scheduling Methods to Grid Environments.* State of the Art and Future Trends, pp. 491 - 506, Kluwer Academic Publishers, 2003.

[11] F. Ygge *Market-Oriented Programming and Its Application to Power Load Management*, PhD thesis, Department of Computer Science, Lund University, 1998.

[12] Rich Wolski, Lawrence J. Miller, Graziano Obertelli, and Martin Swany *Performance Information Services For Computational Grids.* In Resource Management for Grid Computing, Nabrzyski, J., Schopf, J., and Weglarz, J., editors, Kluwer Publishers, Fall, 2003.