Job Allocation Schemes in Computational Grids Based on Cost Optimization

Satish Penmatsa and Anthony T. Chronopoulos [*]
The University of Texas at San Antonio
Dept. of Computer Science
6900 N Loop, 1604 W, San Antonio, Texas 78249, USA
spenmats, atc@cs.utsa.edu

Abstract

In this paper we propose two price-based job allocation schemes for computational grids. A grid system tries to solve problems submitted by various grid users by allocating the jobs to the computing resources governed by different resource owners. The prices charged by these owners are obtained based on a pricing model using a bargaining game theory framework. These prices are then used for job allocation. We present the grid system model and formulate the two schemes as a constraint minimization problem and as a non-cooperative game respectively. The objective of these schemes is to minimize the cost for the grid users. We present algorithms to compute the optimal load (job) fractions to allocate jobs to the computers. Finally, the two schemes are compared under simulations with various system loads and configurations and conclusions are drawn.

1. Introduction

Grid computing [3] is an important developing computing infrastructure which is a conglomeration of computing resources connected by a network, to form a distributed system used for solving complex scientific, engineering and commercial problems. This system tries to solve these problems or applications by allocating the idle computing resources over a network or the internet commonly known as the *computational grid*. These computational resources have different owners who can be enabled by an automated negotiation mechanism by the grid controllers and this can be viewed as a market-oriented grid [2]. This market-oriented grid is concerned with a particular type of resource like the computational power or storage for which these negotiations are made.

A job or an application usually requires the resources from more than one owner. So, these grid computing systems should be able to assign the jobs from various users to the different owned resources efficiently and utilize the resources of unused devices, commonly known as the automated *load balancing/job scheduling* problem. The purpose of load balancing is to improve the performance of the grid system through an appropriate distribution of the user's application load. Formally, this problem can be stated as: given a large number of jobs from various grid users, find the allocation of jobs to computers optimizing a given objective function (e.g.total cost).

Here, we propose two job allocation schemes for a particular grid system model based on an existing pricing model. The two schemes differ in their objective. One tries to minimize the cost of the grid community (i.e. grid users) by taking the jobs at all the grid servers into account whereas the other tries to minimize the cost of the grid users by taking the jobs at each grid server independently of the others. We make simulations and comparisons.

The rest of the paper is organized as follows. In section 2 we review the related works. In section 3 we present the system model and formulate the problem. In section 4 we describe our job allocation schemes and derive allocation algorithms. In section 5 the performance of the schemes are compared. In section 6 we draw conclusions. In section 7 we provide proofs for the stated theorems.

2. Related work

2.1. Pricing model

The grid system is a collection of grid servers and computers (i.e. resources). These servers try to find the resources on which the jobs from various users can be executed. The negotiation between these two entities is formulated as an incomplete information alternating-offer non-cooperative bargaining game in ([9][4][11]) with the grid servers playing on behalf of the grid users. Similar eco-

* This work was supported in part by National Science Foundation under grant CCR-0312323.

nomic models based on game theory are proposed in [1]. The two players (servers and computers) have no idea of each other's reserved valuations [9], i.e. the maximum offered price for the server (acting as the buyer of resource) and the minimum expected price for the computers (acting as the seller of the resource). The server has to play an independent game with each computer associated with it to form the price per unit resource vector, p_j. So, in a system with m servers and n computers at time t, we have $m \times n$ bargaining games as shown in Figure 1.

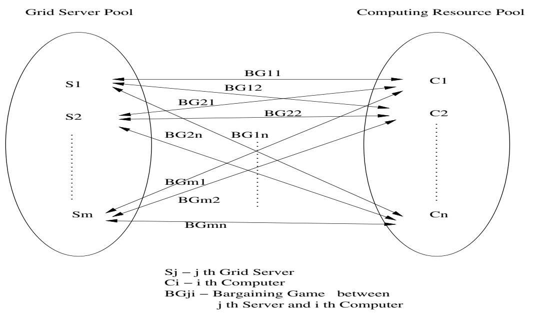

Figure 1. Bargaining game mapping between the grid servers and computers

Both the players try to maximize their utility functions and so the game reduces to the case of dividing the difference of maximum buying price offered by the grid community and minimum selling price expected by the computers.

The bargaining protocol is as follows: One of the players starts the game. If the server starts the game, it proposes an offer which will be much less than its own reserved valuation. If the offered price $\geq$ the computer's standard price with highest expected surplus, then the computer accepts the offer. Else, it makes a counter offer. If this counter offer $\leq$ the server's standard price with the highest expected surplus, it accepts. Else the server counter offers again. This procedure continues until an agreement is reached.

At each step, the expected surplus of each player is based on the probability of acceptance, breakdown or counter-offer of the other player. In general, they are given by: *Expected Utility = E[Surplus] = (reserved valuation of x - standard price of x)×probability(standard price)* [11], where x stands for the grid server or the computer, $probability(standardprice)$ is the probability that the standard price will be accepted by the other player as predicted by itself and the standard price represents the different offered prices used by the players to compute their expected surplus. Also, at each step, if an offer is rejected, then the players will update (i.e. reduce) the $probability(standardprice)$ which monotonically decreases as the alternatives come closer to their reserved valuations where it is more likely to be accepted by the opponent [8].

We simulated this pricing model based on the assumptions given in [4]. Figure's 2 and 3 show the expected surplus (profit) earned by the server and the computer against their various offered prices with time. As the time increases, the expected surplus gradually decreases, which makes both the players offer prices which are much closer to their reserved valuation and which helps the game to converge.

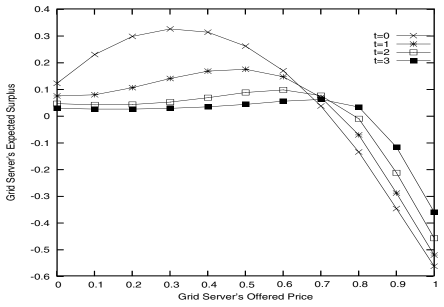

Figure 2. Expected surplus of the Grid server vs Offered prices

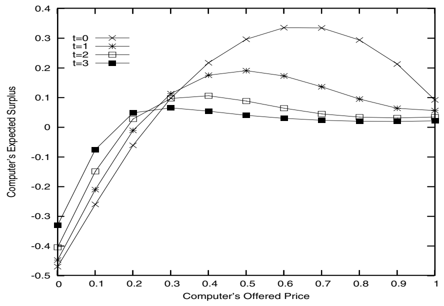

Figure 3. Expected surplus of the Computer vs Offered prices

2.2. Job allocation schemes

2.2.1. The global optimal job allocation scheme we propose in this paper is similar to the one for multi-class model described in [6]. But, in our model, we assume that the jobs assigned to a computer by a grid server are processed completely by itself and are not transferred any further. Based on this, we formulate the problem as a cost minimization problem and provide a solution. Similar work was done in [4]. They considered a grid system model with a single server which accepts each user's jobs and assigns them (taking the pricing into account) to the computers. This is a single-server many-computer scheduling algorithm.

2.2.2. The other scheme is an extension of the NASH distributed load balancing scheme [5] to include pricing where each grid server tries to optimize its objective function (minimizing the cost) independently of the others and they all eventually reach an equilibrium. In general, the jobs from a grid user will be dealt by a local server and so this scheme is favorable to the individual users but not to the entire system. This situation can be viewed as a non-cooperative game among the servers. The equilibrium is called $Nash$ $equilibrium$ and is obtained by a distributed noncooperative policy.

3. System model

We consider a grid system model with many servers and computers as shown in Figure 4. The system has m grid servers and n computers. The grid controller acting on behalf of the grid community (which consists of the users) assigns jobs to the grid servers from different users with a total job arrival rate of Φ. Let the job arrival rate at each server be ϕ_j. Hence, $\Phi = \sum_{j=1}^{m} \phi_j$. Each computer is modeled as an M/M/1 queuing system (i.e. Poisson arrivals and exponentially distributed processing times) [7] and is characterized by its average processing rate μ_i, $i = 1, \ldots, n$. The total job arrival rate Φ must be less than the aggregate processing rate of the system (i.e. $\Phi < \sum_{i=1}^{n} \mu_i$). Each server j keeps track of the price per unit resource p_{ji} (the bargaining game is played prior to the job allocation) and the processing rate μ_i of the i^{th} computer. Since each grid user will have a different reserved valuation, the p_{ji} depends on the user on whose behalf the server j plays the game with the i^{th} computer and so the server has to maintain separate price vectors for each grid user.

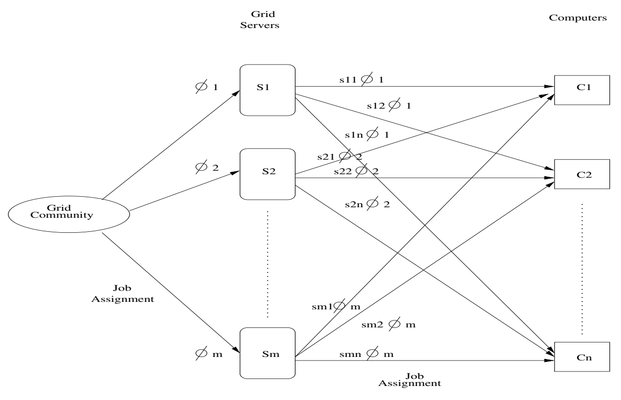

Figure 4. Grid System Model

Based on the scheme, we find the load fractions (s_{ji}) of each server j $(j = 1, \ldots, m)$ that are assigned to computer i ($\sum_{i=1}^{n} s_{ji} = 1$ and $0 \le s_{ji} \le 1$, $i = 1, \ldots, n$) such that the expected price of all the jobs in the system or the expected price of the local jobs of the servers is minimized. In the following we present the notations and define the problem.

Let s_{ji} be the fraction of workload (jobs) that server j sends to computer i. Thus, $\mathbf{s}_j = (s_{j1}, s_{j2}, \ldots, s_{jn})$ denotes the workload fractions of server j and the vector $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m)$ denotes the load fractions of all the servers.

Since each computer is modeled as an M/M/1 queuing system, the expected response time at computer i is given by:

$$F_i(\mathbf{s}) = \frac{1}{\mu_i - \sum_{j=1}^{m} s_{ji}\phi_j} \tag{1}$$

Thus the overall expected cost of server j is given by:

$$D_j(\mathbf{s}) = \sum_{i=1}^{n} k_i p_{ji} s_{ji} F_i(\mathbf{s}) = \sum_{i=1}^{n} \frac{k_i p_{ji} s_{ji}}{\mu_i - \sum_{k=1}^{m} s_{ki}\phi_k} \tag{2}$$

and the overall expected cost of the system (i.e. of all the servers) is given by:

$$D(\mathbf{s}) = \frac{1}{\Phi} \sum_{j=1}^{m} \phi_j D_j(\mathbf{s}) \tag{3}$$

which is equivalent to

$$D(\mathbf{s}) = \frac{1}{\Phi} \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{k_i p_{ji} \phi_j s_{ji}}{\mu_i - \sum_{k=1}^{m} s_{ki}\phi_k} \tag{4}$$

subject to the constraints:

$$s_{ji} \ge 0, \qquad i = 1, \ldots, n, \qquad j = 1, \ldots, m \tag{5}$$

$$\sum_{i=1}^{n} s_{ji} = 1, \qquad j = 1, \ldots, m \tag{6}$$

$$\sum_{j=1}^{m} s_{ji}\phi_j < \mu_i, \qquad i = 1, \ldots, n \tag{7}$$

where k_i is assumed to be a constant which maps the execution time to the amount of resources consumed at node i and p_{ji} is the agreed price as a result of the bargaining game between server j and computer i.

Based on the above we propose two job allocation schemes; $GOSP$ which try to minimize the cost of all the jobs in the system (i.e. jobs at all the servers) and $NASHP$ which try to minimize the cost of all the jobs at each server independently of the others. We describe these two schemes in the next section.

4. Price based job allocation schemes

4.1. Global Optimal Scheme with Pricing ($GOSP$)

The load fractions ($\mathbf{s}$) are obtained by solving the non-linear optimization problem $D(\mathbf{s})$ (4) which gives the optimum expected cost of the system. To find the solution, the scheme finds the load fractions of each server by taking into account the load on each computer due to the other server allocations. Let $\mu_i^j = \mu_i - \sum_{k=1,k\neq j}^{m} s_{ki}\phi_k$ be the *available processing* rate at computer i as seen by server j.

Theorem 1: Assuming that computers are ordered in decreasing order of their available processing rates ($\mu_1^j \geq \mu_2^j \geq \ldots \geq \mu_n^j$), the load fractions for server j are given by:

$$s_{ji} = \begin{cases} \frac{1}{\phi_j}\left(\mu_i^j - \sqrt{k_i p_{ji}\mu_i}\frac{\sum_{k=1}^{c_j}\mu_k^j - \phi_j}{\sum_{k=1}^{c_j}\sqrt{k_k p_{jk}\mu_k}}\right) & \text{if} \quad 1 \leq i < c_j \\ 0 & \text{if} \quad c_j \leq i \leq n \end{cases} \tag{8}$$

where c_j is the minimum index that satisfies the inequality:

$$\mu_{c_j}^j \leq \frac{\sqrt{k_{c_j} p_{jc_j}\mu_{c_j}}(\sum_{k=1}^{c_j}\mu_k^j - \phi_j)}{\sum_{k=1}^{c_j}\sqrt{k_k p_{jk}\mu_k}} \tag{9}$$

Proof: In section 7 (Appendix).

Based on the above theorem we derived the following algorithm for determining server j's best load fractions.

```
BEST-FRACTIONS($\mu_1^j, \ldots, \mu_n^j, \phi_j,$
        $p_{j1}, \ldots, p_{jn}, k_1, \ldots, k_n$)
Input: Available processing rates:
        $\mu_1^j, \mu_2^j, \ldots \mu_n^j$;
    Total arrival rate: $\phi_j$
    The price per unit resource vector:
        $p_{j1}, p_{j2}, \ldots p_{jn}$
    The constants vector: $k_1, k_2, \ldots k_n$
Output: Load fractions: $s_{j1}, s_{j2}, \ldots s_{jn}$;

1. Sort the computers in decreasing order of
   $(\frac{\mu_1^j}{\sqrt{\mu_1 k_1 p_{j1}}} \geq \frac{\mu_2^j}{\sqrt{\mu_2 k_2 p_{j2}}} \geq \ldots \geq \frac{\mu_n^j}{\sqrt{\mu_n k_n p_{jn}}})$;
2. $t \leftarrow \frac{\sum_{i=1}^{n}\mu_i^j - \phi_j}{\sum_{i=1}^{n}\sqrt{\mu_i p_{ji} k_i}}$
3. while ( $t \geq \frac{\mu_n^j}{\sqrt{\mu_n k_n p_{jn}}}$ ) do
        $s_{jn} \leftarrow 0$
        $n \leftarrow n - 1$
        $t \leftarrow \frac{\sum_{i=1}^{n}\mu_i^j - \phi_j}{\sum_{i=1}^{n}\sqrt{\mu_i p_{ji} k_i}}$
4. for $i = 1, \ldots, n$ do
        $s_{ji} \leftarrow \frac{1}{\phi_j}\left(\mu_i^j - t\sqrt{\mu_i p_{ji} k_i}\right)$
```

The following theorem proves the correctness of this algorithm.

Theorem 2: The load fractions $\{s_{j1}, s_{j2}, \ldots, s_{jn}\}$ computed by the BEST-FRACTIONS algorithm solves the optimization problem $D(\mathbf{s})$ and are the optimal fractions for server j.

Proof: In section 7 (Appendix).

To compute the optimal load fractions of all the servers, there should be some communication between them in order to obtain the load information from the other servers and compute the μ_i^j's.

Based on the BEST-FRACTIONS algorithm presented above, we devise the following iterative algorithm where each server updates from time to time its load fractions taking into account the existing load fractions of the other servers in a round-robin fashion.

We use the following notations:

j - the server number;

l - the iteration number;

$\mathbf{s}_j^{(l)}$ - the load fractions of server j computed at iteration l;

$D_j^{(l)}$ - server j's expected price at iteration l;

ϵ - a properly chosen acceptance tolerance;

Send(j, (p, q, r)) - send the message (p, q, r) to server j;

Recv(j, (p, q, r)) - receive the message (p, q, r) from server j;

(where p is a real number, and q, r are integer numbers).

GOSP job allocation algorithm:

```
  User $j$, ($j = 1, \ldots, m$) executes:
1. Initialization:
      $\mathbf{s}_j^{(0)} \leftarrow \mathbf{0}$;
      $\mathbf{D}_j^{(0)} \leftarrow \mathbf{0}$;
      $l \leftarrow 0$;
      $norm \leftarrow 1$;
      $sum \leftarrow 0$;
      $tag \leftarrow$ CONTINUE;
      $left = [(j-2)\text{mod}m] + 1$;
      $right = [j\text{mod}m] + 1$;
2. while ( 1 ) do
      if($j = 1$) {server 1}
          if ($l \neq 0$)
             Recv($left$, ($norm$, $l$, $tag$));
             if ($norm < \epsilon$)
                 Send($right$, ($norm$, $l$, STOP));
                 exit;
          $sum \leftarrow 0$;
          $l \leftarrow l + 1$;
      else {the other servers}
          Recv($left$, ($sum$, $l$, $tag$));
          if ($tag =$ STOP)
```

```
            if ($j \neq m$)
                Send($right$, ($sum$, $l$, STOP));
            exit;
    for $i = 1, \dots, n$ do
            Obtain $\mu_i^j$ by inspecting the run queue
            of each computer
            ($\mu_i^j \leftarrow \mu_i - \sum_{k=1, k \neq j}^{m} s_{ki}\phi_k$);
    $\mathbf{s}_j^{(l)} \leftarrow$ BEST-FRACTIONS($\mu_1^j, \dots, \mu_n^j, \phi_j,$
                                    $p_{j1}, \dots, p_{jn}, k_1, \dots, k_n$);
    Compute $D_j^{(l)}$;
    $sum \leftarrow sum + |D_j^{(l-1)} - D_j^{(l)}|$;
    Send($right$, ($sum$, $l$, CONTINUE));
endwhile
```

This iterative algorithm can be implemented on the distributed system and can be restarted periodically or when the system parameters are changed. Once the accepted tolerance is reached, the servers will continue to use the same load fractions and the system operates at the optimal cost. The running time of each iteration is $O(mnlogn + mnlog(1/\epsilon))$. This Global Optimal Scheme with Pricing ($GOSP$) minimizes the expected cost over all the jobs executed by the Grid system.

4.2. Nash Scheme with Pricing ($NASHP$)

We consider the $NASH$ algorithm for load balancing in distributed systems [5] and modify it to include pricing. In this scheme each server tries to minimize the total cost of its jobs independently of the others. The load fractions are obtained by formulating the problem as a non-cooperative game among the servers. The goal of server j is to find a feasible job allocation strategy $\mathbf{s}_j$ such that $D_j(\mathbf{s})$ (2) is minimized.

The best allocation strategy of server j which is the solution of (2) is given by the following theorem.

Theorem 3: Assuming that computers are ordered in decreasing order of their available processing rates ($\mu_1^j \geq \mu_2^j \geq \dots \geq \mu_n^j$), the solution $\mathbf{s}_j$ of the optimization problem $D_j(\mathbf{s})$ is given by:

$$s_{ji} = \begin{cases} \frac{1}{\phi_j}\left(\mu_i^j - \sqrt{k_i p_{ji} \mu_i^j}\, \frac{\sum_{k=1}^{c_j} \mu_k^j - \phi_j}{\sum_{k=1}^{c_j} \sqrt{k_k p_{jk} \mu_k^j}}\right) & if \quad 1 \leq i < c_j \\ 0 & if \quad c_j \leq i \leq n \end{cases} \tag{10}$$

where c_j is the minimum index that satisfies the inequality:

$$\sqrt{\mu_{c_j}^j} \leq \frac{\sqrt{k_{c_j} p_{jc_j}}\,(\sum_{k=1}^{c_j} \mu_k^j - \phi_j)}{\sum_{k=1}^{c_j} \sqrt{k_k p_{jk} \mu_k^j}} \tag{11}$$

Proof: Similar to that of $GOSP$.

Based on the above theorem we have the **BEST-REPLY** algorithm similar to that of the BEST-FRACTIONS for determining server j's best strategy. The computation of Nash equilibrium may require some communication between the servers. Each server updates from time to time its job allocation strategy by computing the best response against the existing job allocation strategies of the other servers. Based on the BEST-REPLY algorithm we devised a greedy best reply algorithm for computing the Nash equilibrium for the noncooperative job allocation scheme which is similar to the $GOSP$ job allocation algorithm by replacing the procedure call to BEST-FRACTIONS by BEST-REPLY.

5. Experimental results

5.1. Simulation environment

We developed a simulation platform to evaluate the performance of our $GOSP$ and $NASHP$ schemes. The main performance metrics used in our simulations are the *expected response time* and the *fairness index*. The *fairness index* [5] (defined from the servers' perspective),

$$I(\mathbf{C}) = \frac{[\sum_{j=1}^{m} C_j]^2}{m \sum_{j=1}^{m} C_j^2} \tag{12}$$

is used to quantify the fairness of job allocation schemes. Here the parameter $\mathbf{C}$ is the vector $\mathbf{C} = (C_1, C_2, \dots, C_m)$ where C_j is the expected cost of server j's jobs. This index is a measure of the 'equality' of servers' total expected cost. If all the servers have the same total expected price then $I = 1$ and the system is 100% fair to all servers and it is cost-balanced. If the differences on C_j increase, I decreases and the job allocation scheme favors only some servers. If the cost is proportional to the load, then cost-balanced is also load-balanced.

5.2. Performance evaluation

We evaluated the schemes presented above under various system loads and configurations. In the following we present and discuss the simulation results.

5.2.1. Effect of system utilization To study the effect of system utilization we simulated a heterogeneous system consisting of 32 computers with eight different processing rates. This system is shared by 20 servers. The price vector p_j for each server is based on the alternating offer bargaining game previously described. In Table 1, we present the system configuration. The first row contains the relative processing rates of each of the eight computer types. Here, the relative processing rate for computer C_i is defined as the ratio of the processing rate of C_i to the processing rate of

Relative μ_i	1	2	3	4	5	7	8	10
#computers	7	6	5	4	3	3	2	2
μ_i (jobs/sec)	10	20	30	40	50	70	80	100
k_i	1	2	3	4	5	6	7	8

Table 1. System configuration.

the slowest computer in the system. The second row contains the number of computers in the system corresponding to each computer type. The third row shows the processing rate of each computer type in the system. The last row shows the values for k_i, the constant which maps the execution time at the computer i to the amount of resources consumed at i [4].

For each experiment the total job arrival rate in the system Φ is determined by the system utilization ρ and the aggregate processing rate of the system. *System utilization* (ρ) is defined as the ratio of the total arrival rate to the aggregate processing rate of the system:

$$\rho = \frac{\Phi}{\sum_{i=1}^{n} \mu_i} \tag{13}$$

We choose fixed values for the system utilization and determined the total job arrival rate Φ.

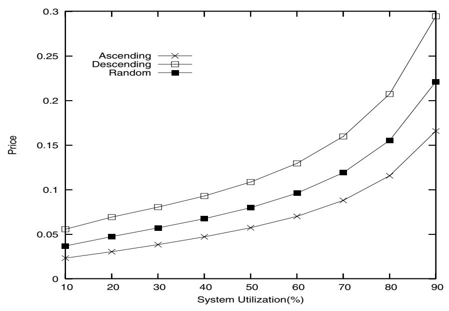

Figure 5. System Utilization vs Expected Price

Figure 5 shows the plots for total price that the grid user has to pay as a function of system utilization based on $GOSP$. The price increases with system utilization because the higher the ρ, the more the load on the computers and so the higher the expected response time and the cost. Three curves are shown corresponding to random, strictly decreasing and strictly increasing price vector (the computers are initially numbered in decreasing order of their processing rates). The random price vector is the one obtained by the pricing strategy described above and the corresponding curve lies between that for the ascending and the descending price vector cases. This is because, if the faster computers charge less (in the case of price vector in ascending order), then they will get the bulk of the work resulting in lower overall response time and subsequently lower total price for the grid user. Similarly, if the faster devices charge more (in the case of price vector in descending order), then they will get fewer jobs resulting in greater overall response time and subsequently greater price for the grid user.

In Figure's 6 and 7, we present the expected response time of the system and the fairness index for different values of system utilization (ranging from 10% to 90%).

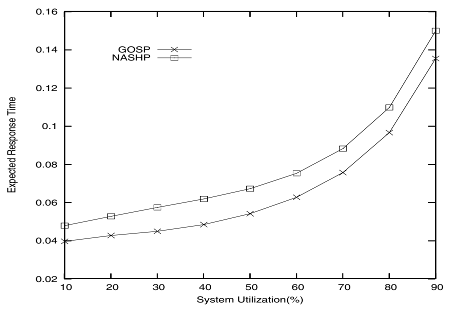

Figure 6. System Utilization vs Expected Response Time

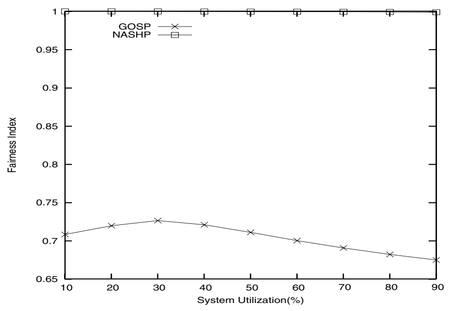

Figure 7. System Utilization vs Fairness Index

It can be seen that for different system loads, the $GOSP$ scheme which minimizes the cost of the entire system performs better than the $NASHP$ scheme where each server minimizes its own cost. But, $GOSP$ whose objective is to reduce the overall cost of the grid community (users) is unfair (fairness index falls to as low as 0.68 for high system loads) whereas $NASHP$ has a fairness index of almost 1 for any system load, which means that it is fair to each server and thus to each user.

5.2.2. Effect of heterogeneity In a grid, heterogeneity usually consists of: processor speed, memory and I/O. A simple way to characterize system heterogeneity is to use the processor speed. Furthermore, it is reasonable to assume that a computer with high speed processor will have matching resources (memory and I/O). One of the common measures of heterogeneity is the *speed skewness* [10] which is defined as the ratio of maximum processing rate to minimum processing rate of the grid computers.

In this section, we investigate the effectiveness of load balancing schemes by varying the speed skewness. We simulate a system of 32 heterogeneous computers: 4 fast and 28 slow. The slow computers have a relative processing rate of 1 and we varied the relative processing rate of the fast computers from 1 (which correspond to a homogeneous system) to 20 (which correspond to a highly heterogeneous system). The system utilization was kept constant $\rho = 60\%$.

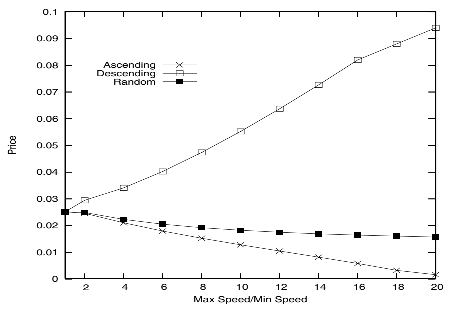

Figure 8. Heterogeneity vs Expected Price

Figure 8 shows the plots for total price that the grid user has to pay with increasing speed skewness for three different price vectors based on $GOSP$. The expected price for the random price vector lies between that for the ascending and descending price vectors for similar reasons as discussed before. Figure 9 plots the overall expected response time with increasing speed skewness. The total response time decreases for both the schemes with an increase in the relative processing speed of the fast computers and $GOSP$ performs better than the $NASHP$ at all times and at high skewness, both perform equally well.

6. Conclusion

In this paper we proposed two job allocation schemes based on pricing for computational grids. These schemes are formulated as a constraint minimization problem and as a non-cooperative game respectively. The algorithms to compute the optimal load (job) fractions for the grid servers are devised. The first scheme tries to minimize the cost of the entire grid system and so is advantageous when the system optimum is required. But it is not fair to the servers and so to the users. The second scheme minimizes the cost for each server. This is fair to the servers and so to the users.

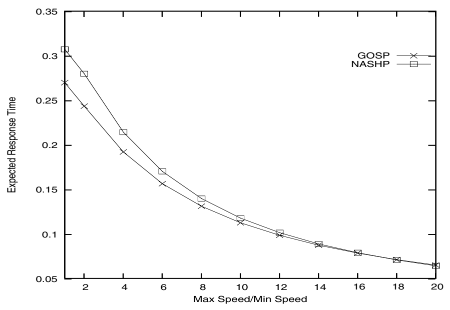

Figure 9. Heterogeneity vs Expected Response Time

7. Appendix

In this section we present the proofs of the results used in the paper.

7.1. Proof of Theorem 1

We begin with the observation that the stability condition (7) is always satisfied because of the fact that the total arrival rate (Φ) does not exceed the total processing rate of the distributed system. Thus we consider $D(\mathbf{s})$ problem with only two restrictions, (5) and (6).

We first show that $D(\mathbf{s})$ is a convex function in $\mathbf{s}$ and that the set of feasible solutions defined by the constraints (5) and (6) is convex.

From (4) it can be easily shown that $\frac{\partial D(\mathbf{s})}{\partial s_{ji}} \geq 0$ and $\frac{\partial^2 D(\mathbf{s})}{\partial (s_{ji})^2} \geq 0$ for $i = 1, \ldots, n$. This means that the Hessian of $D(\mathbf{s})$ is positive which implies that $D(\mathbf{s})$ is a convex function of the load fractions $\mathbf{s}$. The constraints are all linear and they define a convex polyhedron.

Thus, $D(\mathbf{s})$ involves minimizing a convex function over a convex feasible region and the first order Kuhn-Tucker conditions are necessary and sufficient for optimality.

Let $\alpha \geq 0, \eta_{ji} \geq 0, i = 1, \ldots, n, j = 1, \ldots, m$ denote the Lagrange multipliers. The Lagrangian is:

$$L(s_{11}, \ldots, s_{mn}, \alpha, \eta_{11}, \ldots, \eta_{mn}) = \tag{14}$$

$$\sum_{j=1}^{m}\sum_{i=1}^{n} \frac{k_i p_{ji} \phi_j s_{ji}}{\Phi(\mu_i - \sum_{k=1}^{m} s_{ki}\phi_k)} - \alpha(\sum_{j=1}^{m}\sum_{i=1}^{n} s_{ji} - m) - \tag{15}$$

$$\sum_{j=1}^{m}\sum_{i=1}^{n} \eta_{ji} s_{ji} \tag{16}$$

The Kuhn-Tucker conditions imply that s_{ji}, $j = 1, \ldots, m$, $i = 1, \ldots, n$ is the optimal solution to D(**s**) if and only if there exists $\alpha \geq 0$, $\eta_{ji} \geq 0$, $j = 1, \ldots, m, i = 1, \ldots, n$ such that:

$$\frac{\partial L}{\partial s_{ji}} = 0 \quad (17)$$

$$\frac{\partial L}{\partial \alpha} = 0 \quad (18)$$

$$\eta_{ji} s_{ji} = 0, \eta_{ji} \geq 0, s_{ji} \geq 0, j = 1, .., m; i = 1, .., n \quad (19)$$

These conditions become:

$$\frac{k_i p_{ji} \phi_j \mu_i}{\Phi(\mu_i^j - s_{ji}\phi_j)^2} - \alpha - \eta_{ji} = 0, \ \ j = 1, .., m; i = 1, .., n \quad (20)$$

$$\sum_{i=1}^{n} s_{ji} = 1, \qquad j = 1, \ldots, m \quad (21)$$

$$\eta_{ji} s_{ji} = 0, \eta_{ji} \geq 0, s_{ji} \geq 0, j = 1, .., m; i = 1, .., n \quad (22)$$

These are equivalent to:

$$\alpha = \frac{k_i p_{ji} \phi_j \mu_i}{\Phi(\mu_i^j - s_{ji}\phi_j)^2}, \textit{if } s_{ji} > 0; 1 \leq j \leq m; 1 \leq i \leq n \quad (23)$$

$$\alpha \leq \frac{k_i p_{ji} \phi_j \mu_i}{\Phi(\mu_i^j - s_{ji}\phi_j)^2}, \textit{if } s_{ji} = 0; 1 \leq j \leq m; 1 \leq i \leq n \quad (24)$$

$$\sum_{i=1}^{n} s_{ji} = 1, \quad s_{ji} \geq 0; \ \ j = 1, \ldots, m, \ \ i = 1, \ldots, n \quad (25)$$

Claim: Obviously, a computer with a higher average processing rate should have a higher fraction of jobs assigned to it. Under the assumption on the ordering of computers ($\mu_1^j \geq \mu_2^j \geq \ldots \geq \mu_n^j$), we have the following order on load fractions for each server: $s_{j1} \geq s_{j2} \geq \ldots \geq s_{jn}$. This implies that may exist situations in which the slow computers have no jobs assigned to them by the servers. This means that there exist an index c_j ($1 \leq c_j \leq n$) so that $s_{ji} = 0$ for $i = c_j, \ldots, n$ for each server.

From (23) and based on the above claims we can obtain by summation the following equation for each server:

$$\sum_{i=1}^{c_j-1} \sqrt{k_i p_{ji} \phi_j \mu_i} = \sqrt{\alpha}\Phi(\sum_{i=1}^{c_j-1} \mu_i^j - \sum_{i=1}^{c_j-1} s_{ji}\phi_j) \quad (26)$$

Using (24) the above equation becomes:

$$\sqrt{\alpha}\Phi = \frac{\sum_{i=1}^{c_j-1} \sqrt{k_i p_{ji} \phi_j \mu_i}}{\sum_{i=1}^{c_j-1} \mu_i^j - \sum_{i=1}^{c_j-1} s_{ji}\phi_j} \leq \frac{\sqrt{k_i p_{ji} \phi_j \mu_i}}{\mu_{c_j}^j} \quad (27)$$

This is equivalent to:

$$\mu_{c_j}^j \sum_{i=1}^{c_j} \sqrt{k_i p_{ji} \phi_j \mu_i} \leq \sqrt{k_i p_{ji} \phi_j \mu_i}(\sum_{i=1}^{c_j} \mu_i^j - \phi_j) \quad (28)$$

Thus, the index c_j is the minimum index that satisfies the above equation and the result follows.

7.2. Proof of Theorem 2

The while loop in step 3 finds the minimum index c_j for which $\mu_{c_j}^j \leq \frac{\sqrt{k_i p_{ji} \mu_i}(\sum_{k=1}^{c_j} \mu_k^j - \phi_j)}{\sum_{k=1}^{c_j} \sqrt{k_k p_{jk} \mu_k}}$. In the same loop, s_{ji} are set to zero for $i = c_j, \ldots, n$. In step 4, s_{ji} is set equal to $\frac{1}{\phi_j}\left(\mu_i^j - \sqrt{k_i p_{ji} \mu_i} \frac{\sum_{k=1}^{c_j} \mu_k^j - \phi_j}{\sum_{k=1}^{c_j} \sqrt{k_k p_{jk} \mu_k}}\right)$ for $i = 1, \ldots, c_j - 1$. These are in accordance with Theorem 1. Thus, the allocation $\{s_{j1}, \ldots, s_{jn}\}$ computed by the BEST-FRACTIONS algorithm is the optimal solution for each server.

References

[1] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experiance (CCPE), Special Issue on Grid Computing Environments, Wiley Press*, May 2002.

[2] K. M. Chao, R. Anane, J. H. Chen, and R. Gatward. Negotiating agents in a market-oriented grid. In *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, pages 436–437, Berlin, Germany, 2002.

[3] I. Foster and C. Kesselman. *The Grid: Blueprint for a new Computing Infrastructure*. Morgan Kauffman, 2004.

[4] P. Ghosh, N. Roy, K. Basu, and S. Das. A game theory based pricing strategy for job allocation in mobile grids. In *Proc. of the 18th IEEE International Parallel and Distributed Processing Symposium*, pages 26–30, Santa Fe, New Mexico, USA, 2004.

[5] D. Grosu and A. T. Chronopoulos. A game-theoretic model and algorithm for load balancing in distributed systems. In *Proc. of the 16th IEEE International Parallel and Distributed Processing Symposium*, pages 146–153, Ft Lauderdale, Florida, USA, 2002.

[6] H. Kameda, J. Li, C. Kim, and Y. Zhang. *Optimal Load Balancing in Distributed Computer Systems*. Springer Verlag, London, 1997.

[7] L. Kleinrock. *Queueing Systems - Volume 1: Theory*. John Wiley and Sons, 1975.

[8] K. Larson and T. Sandholm. An alternating offers bargaining model for computationally limited agents. *AAMAS'02, Bologna, Italy*, 2002.

[9] G. Owen. *Game Theory*. Academic Press, 1982.

[10] X. Tang and S. T. Chanson. Optimizing static job scheduling in a network of heterogeneous computers. In *Proc. of the International Conf. on Parallel Processing*, pages 373–382, 2000.

[11] P. Winoto, G. McCalla, and J. Vassileva. An extended alternating-offers bargaining protocol for automated negotiation in multi-agent systems. In *Proc. of the 10th International Conference on Cooperative Information Systems*, pages 179–194, Irvine, CA, USA, 2002.