Price-based User-optimal Job Allocation Scheme for Grid Systems

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 a price-based user-optimal job allocation scheme for grid systems whose nodes are connected by a communication network. The job allocation problem is formulated as a noncooperative game among the users who try to minimize the expected cost of their own jobs. We use the concept of Nash equilibrium as the solution of our noncooperative game and derive a distributed algorithm for computing it. The prices that the grid users has to pay for using the computing resources owned by different resource owners are obtained using a pricing model based on a game theory framework. Finally, our scheme is compared with a system-optimal job allocation scheme under simulations with various system loads and configurations and conclusions are drawn.

1. Introduction

A *Grid* [5] is a distributed computing infrastructure which uses the resources of many independent computers connected by a network to solve large-scale computation problems. It can also be viewed as a collection of clusters where the computers in a cluster are connected by a fast local area network. The computational resources in a grid cluster may be under the same administrative domain or can have different resource owners. The grid controllers try to solve the computational problems of the grid users by allocating them to the idle computing resources. These resources which may have different owners can be enabled by an automated negotiation mechanism by the grid controllers.

The grid computing systems should be able to assign efficiently the jobs from various users to the computational resources in a grid which is commonly known as the *job scheduling/load balancing* problem. The purpose of job scheduling is to improve the performance of the grid system through an appropriate distribution of the user's application load.

Past game theory and other related works on job allocation and load balancing in general distributed systems and also in grid systems can be found in [8, 18, 15, 7, 10, 21, 11, 1] and references there in. However, they did not take the communication-subsystem/fairness-to-the-users into account. Here, we consider a grid model where the computers are connected by a communication network. Prior to any job scheduling, the grid controllers submit the users jobs to the various computers based on their resource availability. Then, job scheduling is achieved by transferring some jobs from a heavily loaded computer to a lightly loaded one.

The objective of the job allocation scheme we propose is to minimize the cost of the individual grid users. Since the grid controllers act on behalf of the grid users, we use the term 'grid user' instead of 'grid controller' to avoid ambiguity. The scheme is formulated as a noncooperative game among the grid users who try to minimize the expected cost of their own jobs. The main goal of our job allocation scheme is to provide fairness to all the users, i.e. all the users should have the same expected cost independent of the allocated computer.

In a grid system, to allocate a job to a resource, an agreement should be made between the grid user and the resource owner. We use a pricing model based on the bargaining game theory to obtain the prices charged by the resource owners (computers) [8]. Similar economic models are studied in [2, 20, 16, 17, 3, 4].

*Student Member IEEE

†Senior Member IEEE

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

The two players (users and computers) negotiate until an agreement is reached. Each user has to play an independent game with each computer to obtain its price per unit resource on that computer. We simulated this pricing model and used the prices obtained for job allocation.

The rest of the paper is organized as follows. In section 2, we present the system model. In section 3, we formulate the job allocation problem as a noncooperative game among the grid users. In section 4, the performance of our user-optimal job allocation scheme is compared with a system-optimal job allocation scheme by simulations. Conclusions are drawn in section 5.

2. System Model

We consider a grid system model as shown in Figure 1. The system has n nodes connected by a communication network and is shared by m users. The nodes could be either single computers or clusters (of several computers). The nodes and the communication network are assumed to be product-form queuing network models. In the sequel we use 'computer' and 'node' interchangeably. The terminology and assumptions used similar to [13] are as follows:

- ϕ_i^j : Job arrival rate of user j to computer i.
- ϕ^j : Total job arrival rate of user j.
- $\phi^j = \sum_{i=1}^{n} \phi_i^j$.
- Φ : Total job arrival rate of the system.
- $\Phi = \sum_{j=1}^{m} \phi^j$.
- μ_i : Service rate of computer i.
- β_i^j : Job processing rate (load) of user j at computer i.
- $\beta_i = [\beta_i^1, \beta_i^2, \ldots, \beta_i^m]^T$; $\beta = [\beta_1, \beta_2, \ldots, \beta_n]^T$.
- $\beta^k = [\beta_1^k, \beta_2^k, \ldots, \beta_n^k]^T$.
- x_{rs}^j : Job flow rate of user j from computer r to computer s.
- λ^j : Job traffic through the network of user j.
- $\lambda^j = \sum_{r=1}^{n} \sum_{s=1}^{n} x_{rs}^j$; $\lambda = [\lambda^1, \lambda^2, \ldots, \lambda^m]^T$.
- p_i^j : Price per unit resource on computer i for user j.
- $p^j = [p_1^j, p_2^j, \ldots, p_n^j]^T$.

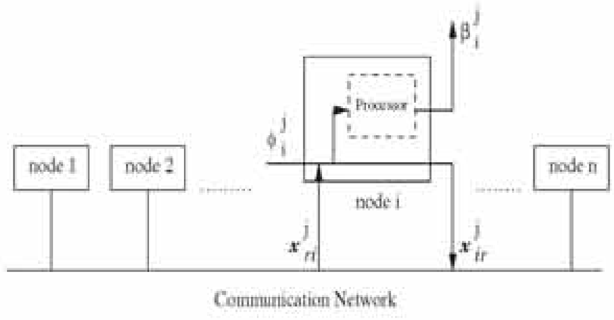

Figure 1. Grid System Model

A job arriving at node i may be either processed at node i or transferred to a neighboring node j for remote processing through the communication network and is not transferred to any further nodes. The decision of transferring a job does not depend on the state of the system and hence is *static* in nature. If a node i sends (receives) jobs to (from) node j, node j does not send (receive) jobs to (from) node i.

For each user j, nodes are classified into the following as in [13]:

- Idle source (R_d^j): does not process any user j jobs ($\beta_i^j = 0$).
- Active source (R_a^j): processes some of the user j jobs that arrive and it sends the remaining user j jobs to other nodes. But, it does not receive any user j jobs from other nodes.
- Neutral (N^j): processes user j jobs locally without sending or receiving user j jobs.
- Sink (S^j): only receives user j jobs from other nodes but it does not send out any user j jobs.

Assuming that each computer is modeled as an M/M/1 queuing system [14], the expected response time of an user j job processed at computer i is given by:

$$F_i^j(\beta_i) = \frac{1}{(\mu_i - \sum_{k=1}^{m} \beta_i^k)} \qquad (1)$$

We assume that the expected communication delay of a job from node r to node s is independent of the source-destination pair (r, s) but may depend on the total traffic through the network denoted by λ where $\lambda = \sum_{j=1}^{m} \lambda^j$. Modeling the communication network as an M/M/1 queuing system [14], the expected communication delay of an user j ($j = 1, \ldots, m$) job is given by:

$$G^j(\lambda) = \frac{t}{(1 - t\sum_{k=1}^{m} \lambda^k)}, \quad \sum_{k=1}^{m} \lambda^k < \frac{1}{t} \qquad (2)$$

where t is the mean communication time for sending and receiving a job form one computer to the other for any user. Note that $F_i^j(\beta_i)$ and $G^j(\lambda)$ are increasing positive functions.

The network traffic of user j can be expressed in terms of the variable β_i^j as:

$$\lambda^j = \frac{1}{2}\sum_{i=1}^{n} |\phi_i^j - \beta_i^j| \quad (3)$$

Thus, the overall expected response time of user j is given by:

$$T^j(\beta) = \frac{1}{\phi^j}\sum_{i=1}^{n} \beta_i^j F_i^j(\beta_i) + \frac{\lambda^j}{\phi^j} G^j(\lambda) \quad (4)$$

$$= \frac{1}{\phi^j}\sum_{i=1}^{n} \frac{\beta_i^j}{(\mu_i - \sum_{k=1}^{m} \beta_i^k)} + \frac{\lambda^j t}{\phi^j(1 - t\sum_{k=1}^{m} \lambda^k)} \quad (5)$$

Let k_i be a constant which maps the execution time to the amount of resources consumed at computer i and let k_c be a constant which maps the communication delay to the amount of resources consumed from the communication network. We assume that the price the users have to pay for consuming a unit resource of the network is constant for all the users and denote it by p_c.

Thus, the overall expected cost of user j is given by:

$$D^j(\beta) = \frac{1}{\phi^j}\sum_{i=1}^{n} \frac{k_i p_i^j \beta_i^j}{(\mu_i - \sum_{k=1}^{m} \beta_i^k)} + \frac{k_c p_c t \lambda^j}{\phi^j(1 - t\sum_{k=1}^{m} \lambda^k)} \quad (6)$$

3. Noncooperative Game among the Users

In this section, we formulate the job allocation problem as a noncooperative game among the users. We use the game theory terminology as in [9]. Each user j $(j = 1, \ldots, m)$ must find the workload (β_i^j) that is assigned to computer i such that the expected price of his own jobs $(D^j(\beta))$ is minimized. The vector $\beta^j = [\beta_1^j, \beta_2^j, \ldots, \beta_n^j]$ is called the job allocation strategy of user j $(j = 1, \ldots, m)$ and the vector $\beta^* = [\beta^1, \beta^2, \ldots, \beta^m]$ is called the strategy profile of the job allocation game. The strategy of user j depends on the job allocation strategies of the other users.

The assumptions for the existence of a feasible strategy profile are as follows:

(i) *Positivity*: $\beta_i^j \geq 0$, $i = 1, \ldots, n$, $j = 1, \ldots, m$;

(ii) *Conservation*: $\sum_{i=1}^{n} \beta_i^j = \phi^j$, $j = 1, \ldots, m$;

(iii) *Stability*: $\sum_{j=1}^{m} \beta_i^j < \mu_i$, $i = 1, \ldots, n$;

A *Noncooperative job allocation game* consists of a set of players, a set of strategies, and preferences over the set of strategy profiles:

(i) *Players*: The m users.

(ii) *Strategies*: Each user's set of feasible job allocation strategies.

(iii) *Preferences*: Each user's preferences are represented by its expected price (D^j). Each user j prefers the strategy profile β^* to the strategy profile $\beta^{*'}$ if and only if $D^j(\beta^*) < D^j(\beta^{*'})$.

We need to solve the above game for our job allocation scheme. A solution can be obtained at the Nash equilibrium [6] which is defined as follows.

Definition 3.1 (Nash equilibrium): *A* Nash equilibrium *of the job allocation game defined above is a strategy profile β^* such that for every user j $(j = 1, \ldots, m)$:*

$$\beta^j \in \arg\min_{\tilde{\beta}^j} D^j(\beta^1, \ldots, \tilde{\beta}^j, \ldots, \beta^m) \quad (7)$$

At the Nash equilibrium, a user j cannot further decrease its expected price by choosing a different job allocation strategy when the other users strategies are fixed. The equilibrium strategy profile can be found when each user's job allocation strategy is a *best response* to the other users strategies.

The *best response* for user j, is a solution to the following optimization problem (BR^j):

$$\min_{\beta^j} D^j(\beta) \quad (8)$$

subject to the constraints:

$$\beta_i^j \geq 0, \qquad i = 1, \ldots, n \quad (9)$$

$$\sum_{i=1}^{n} \beta_i^j = \phi^j \quad (10)$$

$$\sum_{j=1}^{m} \beta_i^j < \mu_i, \qquad i = 1, \ldots, n \quad (11)$$

Remark 3.1 *In finding the solution to BR^j, the strategies of all the other users are kept fixed and so the variables in BR^j are the workloads of user j, i.e. $\beta^j = (\beta_1^j, \beta_2^j, \ldots, \beta_n^j)$.*

In order to solve the optimization problem in eq. (8), we define the following functions:

$$f_i^j(\beta_i) = \frac{\partial}{\partial \beta_i^j}[k_i p_i^j \beta_i^j F_i^j(\beta_i)] = \frac{k_i p_i^j \mu_i^j}{(\mu_i^j - \beta_i^j)^2} \qquad (12)$$

where $\mu_i^j = \mu_i - \sum_{k=1,k\neq j}^{m} \beta_i^k$.

$$g^j(\lambda) = \frac{\partial}{\partial \lambda^j}[k_c p_c \lambda^j G^j(\lambda)] = \frac{k_c p_c t g_{-j}}{(g_{-j} - t\lambda^j)^2} \qquad (13)$$

where $g_{-j} = (1 - t\sum_{k=1,k\neq j}^{m} \lambda^k)$.

$$(f_i^j)^{-1}(\beta_i|_{\beta_i^j=x}) = \begin{cases} (\mu_i^j - \sqrt{\frac{k_i p_i^j \mu_i^j}{x}}), & \text{if } x > \frac{k_i p_i^j}{\mu_i^j} \\ 0, & \text{if } x \leq \frac{k_i p_i^j}{\mu_i^j} \end{cases} \qquad (14)$$

The *best response* strategy of user j, which is the solution of BR^j, is given in the following theorem.

***Theorem* 3.1** *The solution to the optimization problem BR_j satisfies the relations*

$$\begin{array}{ll} f_i^j(\beta_i) \geq \alpha^j + g^j(\lambda), & \beta_i^j = 0 \quad (i \in R_d^j), \\ f_i^j(\beta_i) = \alpha^j + g^j(\lambda), & 0 < \beta_i^j < \phi_i^j \quad (i \in R_a^j), \\ \alpha^j + g^j(\lambda) \geq f_i^j(\beta_i) \geq \alpha^j, & \beta_i^j = \phi_i^j \quad (i \in N^j), \\ f_i^j(\beta_i) = \alpha^j, & \beta_i^j > \phi_i^j \quad (i \in S^j), \end{array} \qquad (15)$$

subject to the total flow constraint,

$$\sum_{i\in S^j} (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j}) + \sum_{i\in N^j} \phi_i^j + \sum_{i\in R_a^j} (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j+g^j(\lambda)}) = \phi^j \qquad (16)$$

where α^j is the Lagrange multiplier.

Proof: The proof uses analogous methodology to [13, 9] and is omitted due to lack of space. □

Since it is not possible to obtain a closed form solution for α^j from eq. (16), we use a binary search to solve eq. (16) iteratively for α^j similar to [13]. Also, the following properties which are true in the optimal solution can be derived from Theorem 3.1 and their proofs are similar to those in [13].

***Property* 3.1**

$$\begin{array}{r} f_i^j(\beta_i|_{\beta_i^j=0}) \geq \alpha^j + g^j(\lambda), \quad \text{iff} \quad \beta_i^j = 0, \\ f_i^j(\beta_i|_{\beta_i^j=\phi_i^j}) > \alpha^j + g^j(\lambda) > f_i^j(\beta_i|_{\beta_i^j=0}), \\ \text{iff} \quad 0 < \beta_i^j < \phi_i^j, \\ \alpha^j \leq f_i^j(\beta_i|_{\beta_i^j=\phi_i^j}) \leq \alpha^j + g^j(\lambda), \quad \text{iff} \quad \beta_i^j = \phi_i^j, \\ \alpha^j > f_i^j(\beta_i|_{\beta_i^j=\phi_i^j}), \quad \text{iff} \quad \beta_i^j > \phi_i^j. \end{array}$$

***Property* 3.2** *If β^j is an optimal solution to the problem in eq. (8) then we have*

$$\begin{array}{l} \beta_i^j = 0, \quad i \in R_d^j, \\ \beta_i^j = (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j+g^j(\lambda)}), \quad i \in R_a^j, \\ \beta_i^j = \phi_i^j, \quad i \in N^j, \\ \beta_i^j = (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j}), \quad i \in S^j. \end{array}$$

***Property* 3.3** *If β^j is an optimal solution to the problem in eq. (8) then we have $\lambda^j = \lambda_S^j = \lambda_R^j$, where*

$$\lambda_S^j = \sum_{i\in S^j}[(f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j}) - \phi_i^j],$$

$$\lambda_R^j = \sum_{i\in R_d^j} \phi_i^j + \sum_{i\in R_a^j}[\phi_i^j - (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j+g^j(\lambda_S)})].$$

Remark 3.2 *The conditions in Property 3.1 help to partition the nodes into one of the four categories mentioned above for user j. Once the node partition for user j is known, his optimal loads can be calculated based on Property 3.2. Property 3.3 states that the job flow out of the sources equals the job flow into the sinks for each user j.*

Based on the above properties, we can have the following definitions (eqs. (17) - (22)) for an arbitrary α^j (≥ 0).

$$S^j(\alpha^j) = \{i | f_i^j(\beta_i|_{\beta_i^j=\phi_i^j}) < \alpha^j\} \qquad (17)$$

$$\lambda_S^j(\alpha^j) = \sum_{i\in S^j(\alpha^j)} [(f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j}) - \phi_i^j] \qquad (18)$$

$$R_d^j(\alpha^j) = \{i | f_i^j(\beta_i|_{\beta_i^j=0}) \geq \alpha^j + g^j(\lambda|_{\lambda^j=\lambda_S^j(\alpha^j)})\} \qquad (19)$$

$$R_a^j(\alpha^j) = \{i | f_i^j(\beta_i|_{\beta_i^j=\phi_i^j}) > \alpha^j + g^j(\lambda|_{\lambda^j=\lambda_S^j(\alpha^j)}) > f_i^j(\beta_i|_{\beta_i^j=0})\} \qquad (20)$$

$$\lambda_R^j(\alpha^j) = \sum_{i\in R_d^j(\alpha^j)} \phi_i^j + \sum_{i\in R_a^j(\alpha^j)} [\phi_i^j - (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j+g^j(\lambda|_{\lambda^j=\lambda_S^j(\alpha^j)})})] \qquad (21)$$

$$N^j(\alpha^j) = \{i | \alpha^j \leq f_i^j(\beta_i|_{\beta_i^j=\phi_i^j}) \leq \alpha^j + g^j(\lambda|_{\lambda^j=\lambda_S^j(\alpha^j)})\} \qquad (22)$$

Thus, if an optimal α^j is given, the node partitions in the optimal solution are characterized as

$$R_d^j = R_d^j(\alpha^j), R_a^j = R_a^j(\alpha^j), N^j = N^j(\alpha^j), \quad S^j = S^j(\alpha^j) \qquad (23)$$

and

$$\lambda^j = \lambda_S^j = \lambda_R^j = \lambda_S^j(\alpha^j) = \lambda_R^j(\alpha^j) \qquad (24)$$

In the following, we present an algorithm for determining user j's best response strategy:

BEST-RESPONSE algorithm:

```
**Input:**   $\phi^j, \beta, \lambda, p^j, \mu_1, \ldots, \mu_n, k_1, \ldots, k_n$.
**Output:**   $\beta^j$.
1. *Initialization:*
      $\beta_i^j \leftarrow \phi_i^j;\ i \in N^j;\ i = 1, \ldots, n$
2. Sort the computers such that
      $f_1^j(\beta_1|_{\beta_1^j=\phi_1^j}) \leq \cdots \leq f_n^j(\beta_n|_{\beta_n^j=\phi_n^j})$
   **If** $f_1^j(\beta_1|_{\beta_1^j=\phi_1^j}) + g^j(\lambda|_{\lambda^j=0}) \geq f_n^j(\beta_n|_{\beta_n^j=\phi_n^j})$,
      STOP (No load balancing is required).
3. Determine $\alpha^j$ (using a binary search):
      $a \leftarrow f_1^j(\beta_1|_{\beta_1^j=\phi_1^j})$
      $b \leftarrow f_n^j(\beta_n|_{\beta_n^j=\phi_n^j})$
      **while**(1) **do**
         $\lambda_S^j(\alpha^j) \leftarrow 0$
         $\lambda_R^j(\alpha^j) \leftarrow 0$
         $\alpha^j \leftarrow \frac{a+b}{2}$
         Calculate: $S^j(\alpha^j), \lambda_S^j(\alpha^j), R_d^j(\alpha^j)$,
         $R_a^j(\alpha^j)$, and $\lambda_R^j(\alpha^j)$ (eqs. (17) - (21))
         in the order given for $i = 1, \ldots, n$
         **If** $(|\lambda_S^j(\alpha^j) - \lambda_R^j(\alpha^j)| < \epsilon)$ **EXIT**
         **If** $(\lambda_S^j(\alpha^j) > \lambda_R^j(\alpha^j))$
            $b \leftarrow \alpha^j$
         **else**
            $a \leftarrow \alpha^j$
4. Determine user $j$'s loads on the computers:
   $\beta_i^j \leftarrow 0$,   for $i \in R_d^j(\alpha^j)$
   $\beta_i^j \leftarrow (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j+g^j(\lambda)})$, for $i \in R_a^j(\alpha^j)$
   $\beta_i^j \leftarrow (f_i^j)^{-1}(\beta_i|_{\beta_i^j=\alpha^j})$,   for $i \in S^j(\alpha^j)$
   $\beta_i^j \leftarrow \phi_i^j$,   for $i \in N^j(\alpha^j)$ (eq. (22))
```

Remark 3.3 *The running time of this algorithm is $O(n \log n + n \log 1/\epsilon)$, where ϵ denotes the tolerance used for computing α^j in step 3 of the algorithm.*

In order to obtain the equilibrium allocation, we need an iterative algorithm where each user updates his strategies (by computing his *best response*) periodically by fixing the other users startegies. We can set a virtual ring topology of the users to communicate and iteratively apply the BEST-RESPONSE algorithm to compute the Nash equilibrium similar to [9].

In the following we present the iterative algorithm (NASHPC) for computing the Nash equilibrium for our noncooperative job allocation game. One of the users can initiate the algorithm (initiating user) who calculates his initial strategies by fixing the other users startegies to zero. An iteration is said to be complete if this *initiating user* receives a message from his left neighbor. He then checks if the error norm is less than a tolerance in which case he sends a terminating message to his right neighbor to be propagated around the ring.

NASHPC distributed job allocation algorithm:

Each user j, $j = 1, \ldots, m$ in the ring performs the following steps in each iteration:

1. Receive the current strategies of all the other users from the left neighbor.

2. If the message is a termination message, then pass the termination message to the right neighbor and EXIT.

3. Update the strategies (β^j) by calling the BEST-RESPONSE

4. Calculate D^j (eq. (6)) and update the error norm.

5. Send the updated strategies and the error norm to the right neighbor.

This algorithm can be restarted periodically or when the system parameters are changed. Users will use the strategies that are computed (at the Nash equilibrium) and the system remains in equilibrium.

4. Experimental Results

We perform simulations to study the impact of system utilization and heterogeneity on the performance of the NASHPC scheme. We also implemented the following job allocation scheme for comparison purposes:

- *Global Optimal Scheme with Pricing and Communication* (**GOSPC**) : This scheme minimizes the expected cost over all the jobs executed by the grid cluster. This is an extension of the *overall optimal scheme* [13] in the multi-class environment to include pricing. The loads (β_i^j) for each user are obtained by solving the following nonlinear optimization problem:

$$\min D(\beta) = \frac{1}{\Phi} \sum_{j=1}^{m} [\sum_{i=1}^{n} k_i p_i^j \beta_i^j F_i^j(\beta_i) + k_c p_c \lambda^j G^j(\lambda)] \tag{25}$$

subject to the constraints:

$$\sum_{i=1}^{n} \beta_i^j = \phi^j \qquad j = 1, \ldots, m \tag{26}$$

$$\beta_i^j \geq 0, \qquad i = 1, \ldots, n; j = 1, \ldots, m \tag{27}$$

Table 1. System configuration.

Relative service rate	1	2	5	10
Number of computers	6	5	3	2
Service rate (jobs/sec)	10	20	50	100
k_i	1	2	3	4

The performance metrics used in our simulations are the *expected response time* and the *fairness index* [12]. The *fairness index* (defined from the users' perspective),

$$I(\mathbf{C}) = \frac{[\sum_{j=1}^{m} C_j]^2}{m \sum_{j=1}^{m} C_j^2} \tag{28}$$

is used to quantify the fairness of job allocation schemes. The parameter $\mathbf{C}$ is the vector $\mathbf{C} = (C_1, C_2, \ldots, C_m)$ where C_j is the expected cost of user j's jobs. If all the users have the same total expected price, then $I = 1$ and the system is 100% fair to all users and it is cost-balanced. If I decreases, then the job allocation scheme favors only some users. In the following we present and discuss the simulation results.

4.1. Effect of system utilization

System utilization (ρ) represents the amount of load on the system. It is defined as the ratio of the total arrival rate to the aggregate service rate of the system:

$$\rho = \frac{\Phi}{\sum_{i=1}^{n} \mu_i} \tag{29}$$

We simulated a heterogeneous system consisting of 16 computers to study the effect of system utilization. The system has computers with four different service rates and is shared by 10 users. The system configuration is shown in Table 1. The first row contains the relative service rates of each of the four computer types. The relative service rate for computer C_i is defined as the ratio of the service rate of C_i to the service rate of 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 service rate of each computer type in the system. The last row shows the values for k_i. It is assigned based on the service rate of the computer [8]. The price vector p^j for each user is obtained based on the alternating offer bargaining game described in section I. k_c and p_c are assumed to be 1 in all the following experiments.

For each experiment the total job arrival rate in the system Φ is determined by the system utilization ρ and the aggregate service rate of the system. The total job arrival rate Φ is chosen by fixing the system utilization. The job arrival rate for each user $\phi^j, j = 1, \ldots, 10$ is determined from the total arrival rate as $\phi^j = q^j \Phi$, where the fractions q^j are given in Table 2. The job arrival rates of each user j, $j = 1, \ldots, 10$ to each computer i, $i = 1, \ldots, 16$, i.e. the ϕ_i^j's are obtained in a similar manner. The mean communication time t is assumed to be 0.001 sec.

Table 2. Job arrival fractions q^j for each user

User	1	2	3-6	7	8-10
q^j	0.3	0.2	0.1	0.07	0.01

In Figure 2, we present the expected response time of the system for different values of system utilization ranging from 10% to 90%. It can be observed that the performance of the NASHPC scheme which minimizes the cost of each user is very close to that of GOSPC which minimizes the cost of the entire system.

Figure 3 shows the fairness index for different values of system utilization. The fairness index of GOSPC varies from 1 at low load, to 0.95 at high load. The NASHPC scheme has a fairness index close to 1 and each user obtains the minimum possible expected price for its own jobs (i.e. it is user-optimal). So, GOSPC scheme whose objective is to reduce the overall cost of the grid cluster is unfair whereas NASHPC is fair to each user.

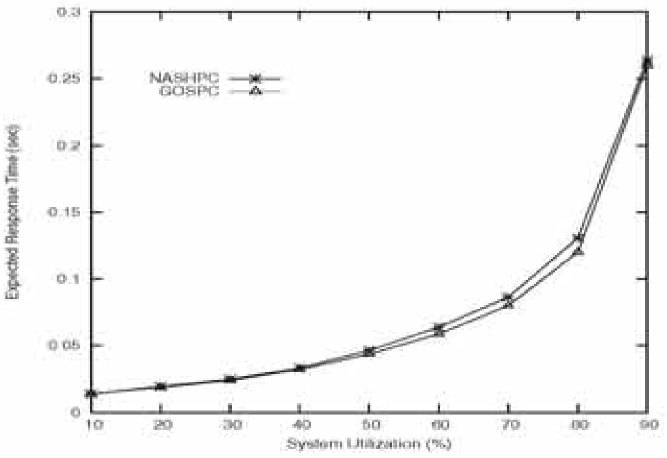

Figure 2. System Utilization vs Expected Response Time

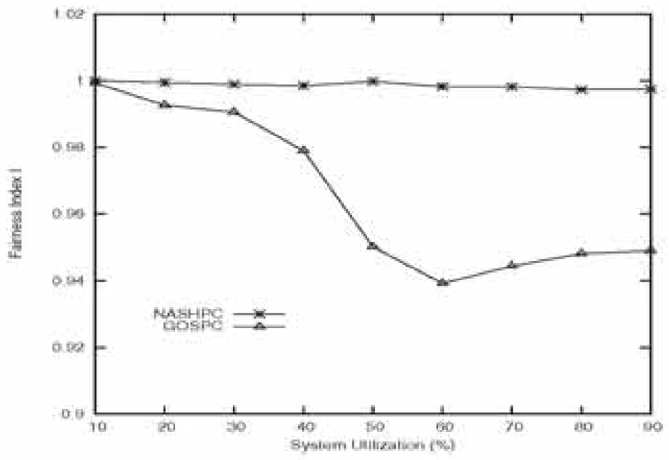

Figure 3. System Utilization vs Fairness Index

4.2. Effect of heterogeneity

Here we study the effect of heterogeneity on the performance of our job allocation scheme. One of the common measures of heterogeneity is the *speed skewness* [19] which is defined as the ratio of maximum service rate to minimum service rate of the grid computers. The effectiveness of NASHPC is studied by varying the speed skewness.

We simulated a heterogeneous system consisting of 16 computers (2 fast and 14 slow) to study the effect of heterogeneity. The slow computers have a relative service rate of 1 and the relative service rate of the fast computers is varied from 1 (homogeneous system) to 20 (highly heterogeneous system). The system utilization was kept constant ($\rho = 60\%$) and the mean communication time t is assumed to be 0.001 sec. In Table 3, we present the service rates (μ_i jobs/sec) of the computers in the systems and the total arrival rates (Φ) for some of the cases. C1 and C2 represent the fast computers and C3 through C16 represent the slow computers. The fractions used to determine the job arrival rate of each user are those presented in Table 2.

Table 3. System parameters

Speed skewness	1	4	8	12	16	20
μ_i of C1,C2	10	40	80	120	160	200
μ_i of C3-C16	10	10	10	10	10	10
Φ (jobs/sec)	96	132	180	228	276	324

Figure 4 shows the effect of speed skewness on the expected response time of the two schemes. It can be observed that, increasing the speed skewness, the NASHPC and GOSPC schemes yield almost the same expected response time which means that in highly heterogeneous grid systems the NASHPC scheme is very effective.

From Figure 5, it can be observed that the fairness index of NASHPC is very close to 1. The fairness index of GOSPC drops from 1 at low skewness to 0.46 at high skewness. This shows that the GOSPC scheme produces an allocation which does not guarantee equal expected price for all the users in the grid cluster. The performance of NASHPC scheme is close to that of GOSPC with the additional advantage of user-optimality.

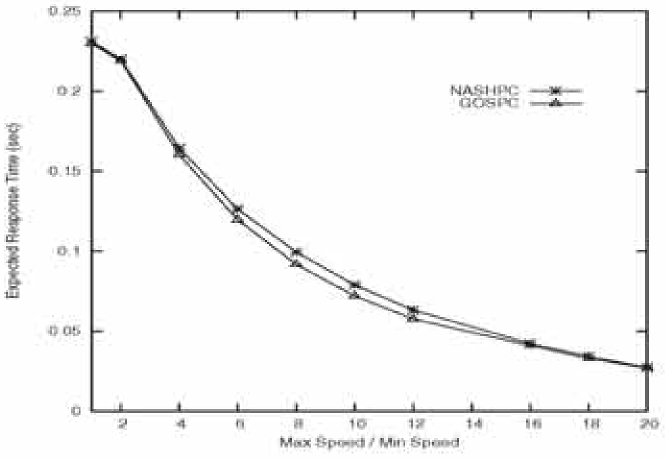

Figure 4. Heterogeneity vs Expected Response Time

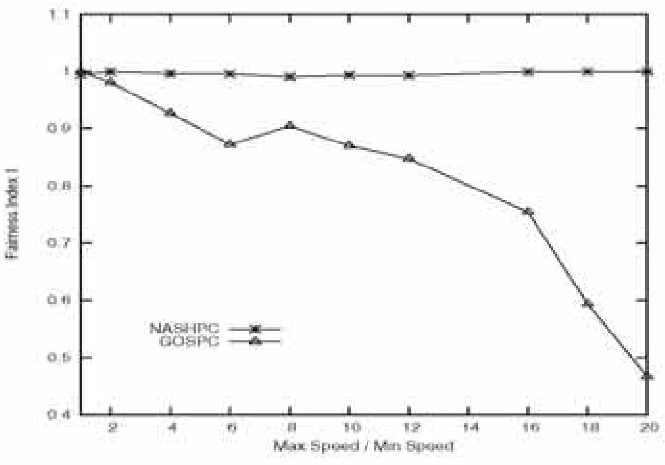

Figure 5. Heterogeneity vs Fairness Index

5. Conclusion

In this paper we proposed a price-based user-optimal job allocation scheme for grid systems. The nodes in the system are connected by a communication network. The job allocation problem is formulated as a noncooperative game among the users whose objective is to minimize the expected cost of their own jobs. We used the Nash equilibrium as the solution of our game and proposed an algorithm to compute the load allocation of the users at the equilibrium. We performed simulations with various system loads and configurations and compared our scheme (NASHPC) with an overall optimal scheme (GOSPC). Based on the simulations, we observed that the performance of the NASHPC scheme is not only comparable with that of GOSPC in terms of the expected response time, but also provides an allocation which is fair (in terms of cost) to all the grid users.

Acknowledgements

Some of the reviewers comments which helped improve the quality of the paper are gracefully acknowledged.

References

[1] E. Altman, H. Kameda, and Y. Hosokawa. Nash equilibria in load balancing in distributed computer systems. *Intl. Game Theory Review*, 4(2):91–100, 2002.

[2] 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.

[3] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. *IEEE Special Issue on Grid Computing*, 93(3):698–714, March 2005.

[4] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal. Scheduling parameter sweep applications on global grids: A deadline and budget constrained cost-time optimisation algorithm. *Intl. Jrnl of Software: Practice and Experiance (SPE)*, 35(5):491–512, April 2005.

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

[6] D. Fudenberg and J. Tirole. *Game Theory.* The MIT Press, 1994.

[7] P. Ghosh, K. Basu, and S. K. Das. Cost-optimal job allocation schemes for bandwidth-constrained distributed computing systems. In *Proceedings of 12th Annual IEEE International Conference on High Performance Computing (HiPC)*, Goa, India, Dec. 2005.

[8] 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.

[9] D. Grosu and A. T. Chronopoulos. Noncooperative load balancing in distributed systems. *Journal of Parallel and Distributed Computing*, 65(9):1022–1034, Sep. 2005.

[10] D. Grosu and A. Das. Auction-based resource allocation protocols in grids. In *Proc. of the 16th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, pages 20–27, Cambridge, Massachusetts, USA, Nov. 2004.

[11] A. Inoie, H. Kameda, and C. Touati. Pareto set, fairness, and nash equilibrium: A case study on load balancing. In *Proc. of the 11th Intl. Symp. on Dynamic Games and Applications*, pages 386–393, Tucson, Arizona, Dec. 2004.

[12] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Wiley-Interscience, 1991.

[13] C. Kim and H. Kameda. Optimal static load balancing of multi-class jobs in a distributed computer system. In *Proc. of the 10th International Conference on Distributed Computing Systems*, pages 562–569, 1990.

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

[15] Y. K. Kwok, S. Song, and K. Hwang. Selfish grid computing: Game-theoretic modeling and nas performance results. In *Proc. of the CCGrid*, Cardiff, U.K., May 2005.

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

[17] M. J. Osborne and A. Rubinstein. *Bargaining and Markets.* Academic Press Inc, 1990.

[18] S. Penmatsa and A. T. Chronopoulos. Job allocation schemes in computational grids based on cost optimization. In *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium, Joint Workshop on HPGC and HIPS*, Denver, Colorado, USA, 2005.

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

[20] 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.

[21] D. Yu and T. G. Robertazzi. Divisible load scheduling for grid computing. In *Proc. of the IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, Marina del Rey, USA, Nov. 2003.