2009 Eighth IEEE International Symposium on Network Computing and Applications

Comparison of Price-based Static and Dynamic Job Allocation Schemes for Grid Computing Systems

Satish Penmatsa
Dept. of Math. & Computer Science
Southern Arkansas University
Magnolia, AR 71753, USA
Email: spenmatsa@saumag.edu

Anthony T. Chronopoulos
Dept. of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249, USA
Email: atc@cs.utsa.edu

***Abstract*—Grid computing systems are a cost-effective alternative to traditional high-performance computing systems. However, the computing resources of a grid are usually far apart and connected by Wide Area Networks resulting in considerable communication delays. Hence, efficient allocation of jobs to computing resources for load balancing is essential in these grid systems. In this paper, two price-based dynamic job allocation schemes for computational grids are proposed whose objective is to minimize the execution cost for the grid users' jobs. One scheme tries to provide a system-optimal solution so that the expected price for the execution of all the jobs in the grid system is minimized, while the other tries to provide a job-optimal solution so that all the jobs in the system of the same size will be charged approximately the same expected price independent of the computers allocated for their execution to provide fairness. The performance of the proposed dynamic schemes is compared with static job allocation schemes using simulations.**

***Index Terms*—job allocation; grid systems; static; dynamic.**

I. Introduction

With the proliferation of computing resources and development of high-speed networks, a *Computational Grid* [1] becomes an important computing infrastructure for large-scale high-performance parallel and distributed applications. These large-scale applications usually require multiple resources which might be under different administrative domains and governed by different resource owners. Also, computational grids often consist of heterogeneous computing resources (computers or nodes) and are usually far apart connected by Wide Area Networks resulting in considerable communication delays. So, a grid system should be able to assign the jobs or applications from various users to the computing resources efficiently so that their execution cost (processing cost + communication cost) is minimized. Hence, efficient job allocation schemes for resource management are pivotal for the operation of a computational grid.

Since the computing resources of a grid are distributed over a larger geographical area and since the load on these resources changes frequently, we here propose job allocation schemes that take the *communication delays* into account and that are *distributed* and *dynamic* in nature. Dynamic schemes make job allocation decisions based on the *current* state of the system. So, they can be more efficient than the static schemes which make job allocation decisions based on the *average* system statistics [2].

The owners of the computing resources in a grid system may charge different prices for executing the grid users' jobs. So, before allocating a job to a resource by the grid controller, an agreement should be made between the grid user and the computer (resource owner). More specifically, an agreement should be made between the software agents [3] of the user and the computer (resource owner). We use a pricing model based on bargaining game theory proposed in [4] to obtain the prices (which are of economic nature) charged by the resource owners. The two players (software agents of the user and the computer) negotiate until an agreement is reached. We simulated this pricing model to obtain the prices required for job allocation.

A. Our Contribution

In this paper, we propose two price-based dynamic job allocation schemes for single-class job computational grids by taking the communication costs into account. We review two static job allocation schemes for conventional distributed systems and extend them to dynamic job allocation schemes for computational grids. The first dynamic scheme (DOPTIM) tries to provide a system-optimal solution so that the expected (mean or average) price (or cost) for the execution of all the jobs in the grid system is minimized. However, some jobs may be charged a higher price than the others even when they are of the same size.

The second dynamic scheme (DCCOOP) tries to provide a job-optimal solution so that all the jobs in the system of approximately the same size will be charged approximately the same expected price independent of the computers allocated for their execution. Jobs of different sizes will be charged relative to their size. Such an allocation is considered *fair* to all the jobs. Fairness of allocation is a major issue in modern utility computing systems such as Amazon Elastic Compute Cloud [5] and Sun Grid Compute Utility [6]. In such systems, users pay the price for the compute capacity they actually consume. Guaranteeing the fairness of allocation to the users in such fixed price settings is an important and difficult problem.

For simplicity and to emphasize our main ideas, in this paper we define *fairness* as follows: If all the jobs of approximately

978-0-7695-3698-9/09 $25.00 © 2009 IEEE
DOI 10.1109/NCA.2009.35

the same size are charged approximately the same expected price independent of the computers allocated for their execution, then we say that the allocation is *fair*. However, in real distributed systems, jobs will rarely be of the same size. We note that our schemes can be easily extended to consider jobs of different sizes.

The performance of the static and the dynamic job allocation schemes is compared using simulations.

B. Past Related Work

Load balancing/job allocation in conventional distributed systems with the objective of minimizing the expected response time (average execution time) of the jobs in the system has been extensively studied. For example, in [7], [8], [9], [10], [11], [12], [13], [14] and references there-in. In grid computing systems, since the computing resources may be managed by different owners who charge different prices, job allocation schemes which minimize the expected price (or cost) for the grid users' jobs are also essential.

Job allocation for resource management in grid computing systems based on different economic and system models has been studied in [15], [16], [17], [4], [18] and references there-in. However, most of the above schemes for computational grids considered the optimization of the overall system cost (did not consider fairness) and the communication costs were not taken into account. Load balancing in computational grids based on cooperative and non-cooperative game theory was studied in [19], [20], [21] and references there-in. However, either the communication subsystem was not taken into account or pricing was not included in the models or the schemes are static in nature. A static price-based job allocation scheme for mobile grid systems was studied in [4]. Its objective is to provide a system-optimal solution and the communication costs were not taken into account. Price-based job allocation for computational grids by taking the communication costs into account was studied in [22]. However, the proposed scheme is static in nature. Adaptive load balancing in computational grids based on load estimation was studied in [23]. However, the proposed scheme does not optimize the cost for the grid users. A system optimal dynamic job allocation scheme for multi-class job grid systems has been studied in [24]. However, it does not provide fairness to the grid users' jobs.

C. Organization

The rest of the paper is organized as follows: In Section II, we present the grid system model. In Section III, two static job allocation schemes for conventional distributed systems are reviewed. In Section IV, two price-based dynamic job allocation schemes for computational grids are proposed. The static and dynamic job allocation schemes are compared using simulations in Section V. Conclusions are drawn in Section VI.

II. Computational Grid System Model

We consider a computational grid system model as shown in Figure 1. The system has 'n' nodes (computers or resources) ($C_i, i = 1, \ldots, n$) connected by a communication network. The nodes could be either single computers or clusters (of several computers). The computers (nodes) and the communication network are modeled as M/M/1 queuing systems [25]. In these queuing systems, the inter-arrival times and the service times are exponentially distributed.

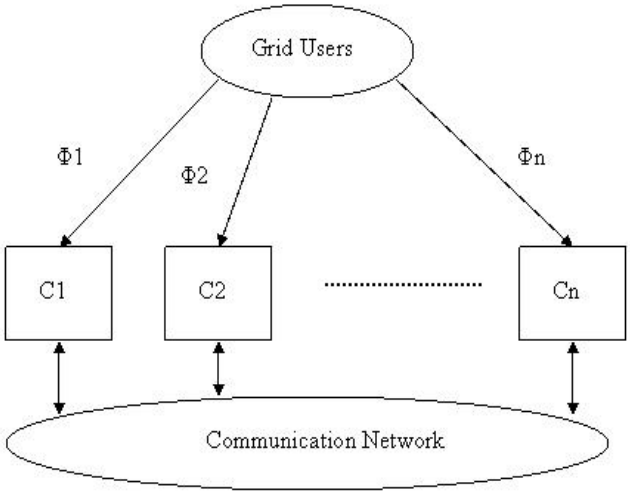

Fig. 1. Grid System Model

The terminology and assumptions used are as follows:

- ϕ_i : External job arrival rate at node i. A job arriving at node i may be either processed at node i or transferred to another node j through the communication network for remote processing.
- Φ : Total external job arrival rate of the system. So, $\Phi = \sum_{i=1}^{n} \phi_i$.
- μ_i : Mean service rate of node i.
- r_i : Mean service time of a job at node i (*i.e.* the average time to service (process) a job at node i).
- β_i : Mean job processing rate (load) at node i (*i.e.* the average number of jobs processed at node i per unit interval of time). This is the load for node i assigned by the job allocation scheme.
- λ : Total traffic through the network.
- t : Mean communication time for sending or receiving a job from one node to another.
- ρ : Utilization of the communication network ($\rho = t\lambda$).

III. Static Job Allocation

In this section, we review two static job allocation schemes for conventional distributed systems, OPTIM [7] and CCOOP [26], based on which two price-based dynamic job allocation schemes for computational grids will be derived in the next section.

The response time of a job in a system as above consists of a node delay (queuing delay + processing delay) at the processing node and also some possible communication delay incurred due to a job transfer. Based on our assumptions on

the node and network models, the mean node delay of a job at node i is given by:

$$F_i(\beta_i) = \frac{1}{\mu_i - \beta_i}, \quad i = 1, \ldots, n. \tag{1}$$

and the mean communication delay for a job is given by:

$$G(\lambda) = \frac{t}{1 - t\lambda}, \quad \lambda < \frac{1}{t} \tag{2}$$

A. Overall Optimal Scheme (OPTIM)

The static overall optimal scheme (OPTIM) [7], [12] determines a load allocation which minimizes the expected response time of jobs in the whole system *i.e.*, its objective is to provide a system-optimal solution. The problem of minimizing the entire system expected job response time is expressed as:

$$\min D(\beta) = \frac{1}{\Phi}[\sum_{i=1}^{n} \beta_i F_i(\beta_i) + \lambda G(\lambda)] \tag{3}$$

subject to the constraints:

$$\sum_{i=1}^{n} \beta_i = \Phi \tag{4}$$

$$\beta_i \geq 0, \qquad i = 1, \ldots, n \tag{5}$$

The above non-linear optimization problem is solved by using the Kuhn-Tucker theorem and an algorithm to compute the optimal loads (β_i, $i = 1, \ldots, n$) is presented in [7], [12].

The marginal node delay $f_i(\beta_i)$ and marginal communication delay $g(\lambda)$ which are used in providing a solution to the problem in eq. (3) are defined as:

$$f_i(\beta_i) = \frac{\partial}{\partial \beta_i} \beta_i F_i(\beta_i) = \frac{\mu_i}{(\mu_i - \beta_i)^2} \tag{6}$$

$$g(\lambda) = \frac{\partial}{\partial \lambda} \lambda G(\lambda) = \frac{t}{(1 - t\lambda)^2} \tag{7}$$

where $\beta_i F_i(\beta_i)$ denotes the mean number of jobs at node i and $\lambda G(\lambda)$ denotes the mean number of jobs in the communication network. The marginal node delays (f_i, $i = 1, \ldots, n$) among the nodes are balanced when the overall optimum is realized.

B. Cooperative Scheme (CCOOP)

The static cooperative scheme (CCOOP) [26] determines a load allocation that provides fairness to all the jobs in the system *i.e.*, all the jobs of the same size will experience the same expected response time independent of the computers allocated for their execution.

The job allocation problem is formulated as a cooperative game among the computers and the communication subsystem. The several decision makers (e.g., computers and the communication subsystem) cooperate in making decisions such that each of them will operate at its optimum. Based on the *Nash Bargaining Solution* (NBS) [26] which provides a Pareto optimal [26] and fair solution, an algorithm is provided for computing the NBS for the cooperative job allocation game.

The following definitions are used in providing the solution:

$$\tilde{f}_i(\beta_i) = \frac{\partial}{\partial \beta_i} \ln F_i(\beta_i) = \frac{1}{\mu_i - \beta_i} = F_i(\beta_i) \tag{8}$$

$$\tilde{g}(\lambda) = \frac{\partial}{\partial \lambda} \ln G(\lambda) = \frac{t}{(1 - t\lambda)} = G(\lambda) \tag{9}$$

The expected node delays ($\tilde{f}_i$, $i = 1, \ldots, n$) among the nodes are balanced when the cooperative solution is realized.

IV. Price-based Dynamic Job Allocation

In this section, we propose two price-based distributed dynamic job allocation schemes for computational grids based on the static schemes presented in the previous section. The objective of these dynamic schemes is to minimize the cost for the grid users' jobs *i.e*, to minimize the expected price for the execution of the jobs on the grid resources. We follow an approach similar to [9], [11].

A distributed dynamic scheme has three components: 1) an *information policy* used to disseminate load information among nodes, 2) a *transfer policy* that determines whether job transfer activities are needed by a node, and 3) a *location policy* that determines the nodes that are suitable to participate in load exchanges.

The dynamic schemes which we propose use the number of jobs waiting in the queue to be processed (queue length) as the state information. The state information is exchanged between the nodes periodically. When a job arrives at a node, the transfer policy component determines whether the job should be processed locally or should be transferred to another node for remote processing. If the job is eligible for transfer, the location policy component determines the destination node for remote processing. We assume that a job can be transferred more than once.

Following are some additional notations for dynamic allocation:

- N_i : Mean number of jobs at node i.
- n_i : Number of jobs at node i at a given instant.
- $\rho^{'}$: Utilization of the communication network at a given instant.

A. Price-based Dynamic Overall Optimal Scheme (DOPTIM)

The objective of DOPTIM is to balance the workload among the nodes dynamically in order to provide a system-optimal solution *i.e.*, to minimize the expected price for the execution of all the jobs over the entire grid system. We base the derivation of DOPTIM on the static OPTIM described in Section III-A.

In the following, we express the marginal node delay at node i ($f_i(\beta_i)$ in eq. (6)) in terms of the mean service time at node i (*i.e.* r_i) and the mean number of jobs at node i (*i.e.* N_i).

Using the relation $\mu_i = \frac{1}{r_i}$ [25], eq. (1) can be written as:

$$F_i(\beta_i) = \frac{1}{\mu_i - \beta_i} = \frac{r_i}{1 - r_i \beta_i} \tag{10}$$

Using Little's Law [25] (*i.e.* $N_i = \beta_i F_i(\beta_i)$), the above equation can be written as:

$$F_i(\beta_i) = r_i(N_i + 1) \tag{11}$$

By taking the partial derivative of $\beta_i F_i(\beta_i)$ *w.r.t.* β_i, we have

$$f_i(\beta_i) = \frac{\partial}{\partial \beta_i}\beta_i F_i(\beta_i) = \frac{\partial}{\partial \beta_i}\Big(\frac{\beta_i r_i}{1 - r_i\beta_i}\Big)$$

Thus, we have

$$f_i(\beta_i) = \frac{r_i}{(1 - r_i\beta_i)^2}$$

Applying Little's law to the above expression gives

$$f_i(\beta_i) = \frac{r_i}{\Big(1 - \frac{r_i N_i}{F_i(\beta_i)}\Big)^2}$$

Using eq. (11),

$$f_i(\beta_i) = \frac{r_i}{\Big(1 - \frac{r_i N_i}{r_i(N_i+1)}\Big)^2} = r_i(N_i + 1)^2.$$

Therefore,

$$f_i(\beta_i) = r_i(N_i + 1)^2 \tag{12}$$

Rewriting eq. (7) in terms of ρ, we have

$$g(\lambda) = \frac{t}{(1 - \rho)^2}, \quad \rho < 1 \tag{13}$$

Let p_i denote the price per unit resource at node i, k_i denote a constant which maps the response time to the amount of resources consumed at node i, p_c denote the price for consuming a unit resource of the communication network, and k_c denote a constant which maps the communication delay to the amount of resources consumed from the communication network. We note that p_i $(i = 1, \ldots, n)$ and p_c will be obtained based on the bargaining game described in [4] played between the software agents of the grid user and the computer (resource owner) before the job allocation is made. We now convert the above marginal node and communication delays into marginal node and communication costs using p_i, k_i, p_c, and k_c.

The marginal node and communication costs can be expressed as:

$$f_i(\beta_i, p_i) = k_i p_i r_i(N_i + 1)^2 \tag{14}$$

$$g(\lambda, p_c) = \frac{k_c p_c t}{(1 - \rho)^2}, \quad \rho < 1 \tag{15}$$

Expressing $f_i()$ in eq. (14) and $g()$ in eq. (15) which use the mean estimates of the system parameters in terms of instantaneous variables, we have, the marginal virtual node cost as:

$$f_i = k_i p_i r_i(n_i + 1)^2 \tag{16}$$

and the marginal virtual communication cost as:

$$g = \frac{k_c p_c t}{(1 - \rho')^2}, \quad \rho' < 1 \tag{17}$$

where n_i is the number of jobs at node i at a given instant and ρ' is the utilization of the communication network at a given instant.

The above relations (eqs. (16) and (17)) are used as the estimates of the marginal node and communication costs. Whereas OPTIM tries to balance the marginal node delays at all the nodes statically, DOPTIM will be derived to balance the marginal virtual node costs at all the nodes dynamically. For a job arriving at node i that is eligible to transfer, each potential destination node j $(j = 1, \ldots, n; j \neq i)$ is compared with node i.

***Definition* 4.1:** *If $f_i > f_j + g$, then it is more costly for a job to be executed at node i than if transferred to node j.*

We use the following proposition to determine a less costly node j relative to node i for a job, in the discussion below.

***Proposition* 4.1:** *Let*

$$n_{ij} = \Big[\frac{k_j p_j r_j}{k_i p_i r_i}(n_j + 1)^2 + \frac{g}{k_i p_i r_i}\Big]^{1/2} - 1. \tag{18}$$

If $n_i > n_{ij}$, then node i is more costly than node j for executing a job.

Proof: It is more costly for a job to be executed at node i than if transferred to node j if $f_i > f_j + g$. Substituting eq. (16) for f_i and f_j, we have:

$$k_i p_i r_i(n_i + 1)^2 > k_j p_j r_j(n_j + 1)^2 + g$$

which is equivalent to

$$(n_i + 1)^2 > \frac{k_j p_j r_j}{k_i p_i r_i}(n_j + 1)^2 + \frac{g}{k_i p_i r_i}$$

Thus, we have,

$$n_i > \Big[\frac{k_j p_j r_j}{k_i p_i r_i}(n_j + 1)^2 + \frac{g}{k_i p_i r_i}\Big]^{1/2} - 1$$

Replacing the right-hand side of the above equation by n_{ij}, we have $n_i > n_{ij}$ where

$$n_{ij} = \Big[\frac{k_j p_j r_j}{k_i p_i r_i}(n_j + 1)^2 + \frac{g}{k_i p_i r_i}\Big]^{1/2} - 1$$

□

The description of the components of DOPTIM is as follows:

(1) ***Information policy*:** Each node i $(i = 1, \ldots, n)$ broadcasts the number of jobs in its queue (*i.e.*, n_i) to all the other nodes. This state information exchange is done periodically, say every P time units.

(2) ***Transfer policy*:** A *threshold policy* is used to determine whether an arriving job should be processed locally or should be transferred to another node. A job arriving at node i will be eligible to transfer when the number of jobs at node i is greater than some threshold denoted by T_i. Otherwise the job will be processed locally. The threshold T_i at a node i $(i = 1, \ldots, n)$ is calculated as follows:

The threshold of the fastest node h, T_h, is chosen as the basis based on its processing speed. For node i ($i = 1, \ldots, n$, $i \neq h$), substituting eq. (16) in the equation $f_i = f_h$, we have:

$$n_i = \sqrt{\frac{k_h p_h r_h}{k_i p_i r_i}}(n_h + 1) - 1$$

Replacing n_i and n_h in the above equation by T_i and T_h respectively, the threshold for a node i can be calculated using the following equation:

$$T_i = \sqrt{\frac{k_h p_h r_h}{k_i p_i r_i}}(T_h + 1) - 1 \tag{19}$$

We note that T_i should be non-negative and an integer.

(3) ***Location policy*:** The destination node for a job at node i that is eligible to transfer is determined based on the state information that is exchanged from the information policy. First, a node with the least marginal virtual cost is determined. Next, it is determined whether the arriving job should be transferred based on the transfers the job already had.

(i) A least costly (least expensive) node for a job is determined as follows: From Proposition 4.1, we have that if $n_i > n_{ij}$, then node i is more costly than node j for executing a job. Else, if $n_i \leq n_{ij}$ then node i is less costly than node j. Let $\delta_{ij} = n_i - n_{ij}$ and $\delta_i = \max_j \delta_{ij}$. If $\delta_i > 0$, then node j is the least costly node for a job. Else, no less costly node is found and the job will be processed locally.

(ii) Let c denote the number of times that a job has been transferred. Let ω ($0 < \omega \leq 1$) be a *weighting factor* used to prevent a job from being transferred continuously and let Δ ($\Delta > 0$) be a *bias* used to protect the system from instability by forbidding the job allocation policy to react to small cost distinctions between the nodes. If $(\omega)^c \delta_i > \Delta$, then the job will be transferred to node j. Otherwise, it will be processed locally.

B. Price-based Dynamic Cooperative Scheme (DCCOOP)

The objective of DCCOOP is to balance the workload among the nodes dynamically in order to obtain a job-optimal solution *i.e.*, all the jobs of approximately the same size should be charged approximately the same expected price independent of the computers allocated for their execution. We base the derivation of DCCOOP on the static CCOOP described in Section III-B.

Expressing $\tilde{f}_i(\beta_i)$ (node delay) in eq. (8)) in terms of the mean service time at node i (*i.e.* r_i) and the mean number of jobs at node i (*i.e.* N_i), and $\tilde{g}(\lambda)$ (communication delay) in eq. (9) in terms of ρ, we have:

$$\tilde{f}_i(\beta_i) = r_i(N_i + 1) \tag{20}$$

$$\tilde{g}(\lambda) = \frac{t}{(1-\rho)}, \quad \rho < 1 \tag{21}$$

Converting the above node and communication delays (eqs. (20) and (21)) into virtual node and communication costs, we have:

$$\tilde{f}_i(\beta_i, p_i) = k_i p_i r_i (N_i + 1) \tag{22}$$

$$\tilde{g}(\lambda, p_c) = \frac{k_c p_c t}{(1-\rho)}, \quad \rho < 1 \tag{23}$$

Expressing the above node and communication costs (eqs. (22) and (23)) which use the mean estimates of the system parameters in terms of instantaneous variables, we have, the virtual node cost as:

$$\tilde{f}_i = k_i p_i r_i (n_i + 1) \tag{24}$$

and the virtual communication cost as:

$$\tilde{g} = \frac{k_c p_c t}{(1-\rho')}, \quad \rho' < 1 \tag{25}$$

The above relations (eqs. (24) and (25)) are used as the estimates of the node and communication costs. Whereas CCOOP tries to balance the node delays at all the nodes statically, DCCOOP will be derived to balance the virtual node costs at all the nodes dynamically. For a job arriving at node i that is eligible to transfer, each potential destination node j ($j = 1, \ldots, n; j \neq i$) is compared with node i.

***Definition* 4.2:** *If $\tilde{f}_i > \tilde{f}_j + \tilde{g}$, then it is more costly for a job to be executed at node i than if transferred to node j.*

We use the following proposition to determine a less costly node j relative to node i for a job.

***Proposition* 4.2:** *Let*

$$n_{ij} = \left\lceil \frac{k_j p_j r_j}{k_i p_i r_i}(n_j + 1) + \frac{\tilde{g}}{k_i p_i r_i} \right\rceil - 1. \tag{26}$$

If $n_i > n_{ij}$, then node i is more costly than node j for executing a job.

Proof: Similar to Proposition 4.1. □

The description of the components of DCCOOP (information policy, transfer policy and location policy) is similar to that of the components of DOPTIM.

- The threshold T_i at a node i ($i = 1, \ldots, n$) is calculated as follows:
  The threshold of the fastest node h, T_h, is chosen as the basis based on its processing speed. For node i ($i = 1, \ldots, n$, $i \neq h$), substituting eq. (24) in the equation $\tilde{f}_i = \tilde{f}_h$, we have:

  $$n_i = \frac{k_h p_h r_h}{k_i p_i r_i}(n_h + 1) - 1$$

  Replacing n_i and n_h in the above equation by T_i and T_h respectively, the threshold for a node i can be calculated using the following equation:

  $$T_i = \frac{k_h p_h r_h}{k_i p_i r_i}(T_h + 1) - 1 \tag{27}$$

  We note that T_i should be non-negative and an integer.
- A least costly (least expensive) node for a job that is eligible to transfer is determined as follows: From Proposition 4.2, we have that if $n_i > n_{ij}$, then node i is more costly than node j for executing a job. Else, if $n_i \leq n_{ij}$ then node i is less costly than node j. Let

$\delta_{ij} = n_i - n_{ij}$ and $\delta_i = \max_j \delta_{ij}$. If $\delta_i > 0$, then node j is the least costly node for a job. Else, no less costly node is found and the job will be processed locally.
Similar to DOPTIM, a *weighting factor* is used to prevent a job from being transferred continuously and a *bias* is used to protect the system from instability by forbidding the job allocation policy to react to small cost distinctions between the nodes.

V. Experimental Results

In this section, the performance of the proposed dynamic job allocation schemes (DOPTIM and DCCOOP) is evaluated using a 16 node heterogeneous grid system model. The Sim++ [27] simulation software package is used to obtain the data and parameters for the experiments. We also implemented the Overall Optimal scheme (OPTIM) [7], the Cooperative scheme (CCOOP) [26], the Proportional scheme (PROP) [28], and Price-based Optimal Allocation scheme (PRIMANGLE) [29] for comparison. PROP allocates the jobs in proportion to the processing speeds of the computers. PRIMANGLE is a static scheme which is an extension of OPTIM to include pricing. The overhead (OV) for job transfer used in the discussion below is defined as the percentage of service time that a computer has to spend to send or receive a job.

The grid system model has computers with four different service rates. The system configuration is shown in Table I. 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, the constant which maps the response time to the amount of resources consumed at computer i. They are assigned based on the service rate of the computers as in [4]. The performance metrics used in our experiments are the *expected response time* and the *expected price* (or *expected cost*).

TABLE I
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

A. Effect of System Utilization

Figure 2 presents the effect of system utilization (system load) ranging from 10% to 90% on the performance of the job allocation schemes when the overhead for job transfer (OV) is 0. It can be observed that at low system utilization, all the schemes except PROP show similar performance. The poor performance of PROP is because the less powerful computers are significantly overloaded. However, as the utilization increases, the dynamic schemes, DOPTIM and DCCOOP show superior performance compared to OPTIM, CCOOP, and PROP. This is because, current system state information is used by the dynamic schemes and the jobs are transferred more effectively to the computers which minimize the response time. DOPTIM, whose objective is to provide a system-optimal solution has a lower expected response time than DCCOOP whose objective is to provide a job-optimal solution.

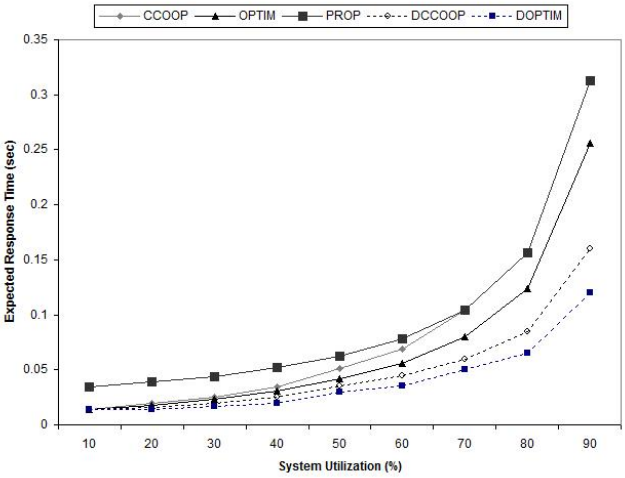

Fig. 2. Mean Response Time vs System Util. (OV = 0)

Figure 3 shows the expected price (in some monetary unit) at each computer i ($i = 1, \ldots, 16$) for a system utilization of 80% and when there is no overhead for job transfer. It can be observed that the expected price in the case of DOPTIM and DCCOOP is very much lower when compared to that of PRIMANGLE at all the computers. Also, the expected price at all the computers is approximately the same in the case of DCCOOP. However, in the case of PRIMANGLE and DOPTIM, some computers have a higher expected price than the others. In the case of DCCOOP, all the jobs are charged approximately the same expected price independent of the computers allocated for their execution. Thus, the allocation provided by DCCOOP can be considered fair. But, in the case of DOPTIM, some jobs may be charged a higher price than the others for their execution and so may not be fair. DOPTIM has the lowest overall expected response time and the lowest overall expected price and so is beneficial when the system-optimum is desired. Although the expected prices at each computer for $\rho = 80\%$ are only shown, similar behavior of the various schemes is observed for any system utilization.

Figure 4 presents the effect of system utilization on the performance of the job allocation schemes when OV = 5%. It can be again observed that at low system utilization, all the schemes except PROP show similar performance. As the

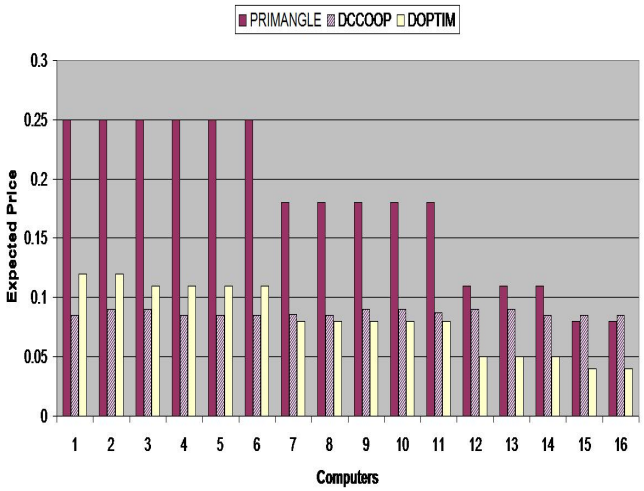

Fig. 3. Expected Price at Computers (OV = 0)

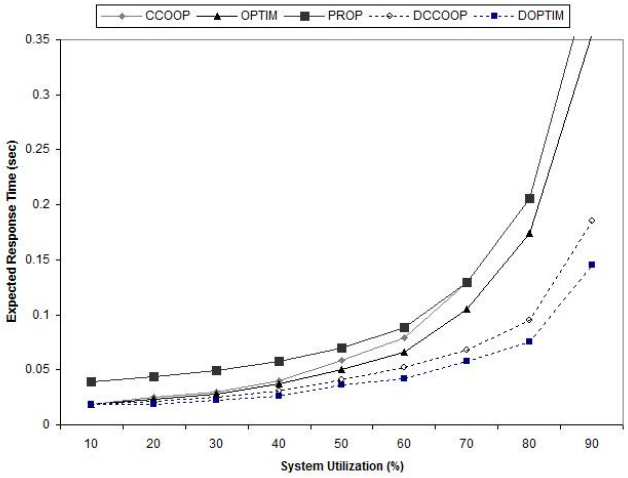

Fig. 4. Mean Response Time vs System Util. (OV = 5%)

utilization increases, the expected response time for all the schemes increases. However, both DOPTIM and DCCOOP still show superior performance compared to OPTIM, CCOOP, and PROP. DOPTIM tries to provide a system-optimal solution and so has a lower expected response time than DCCOOP.

Figure 5 shows the expected price at each computer i ($i = 1, \ldots, 16$) when the system utilization is 80% and OV = 5%. It can be again observed that the expected price in the case of DOPTIM and DCCOOP is very much lower when compared to PRIMANGLE. Also, the expected price at all the computers is approximately the same in the case of DCCOOP but there are large differences in the case of PRIMANGLE and DOPTIM. Thus, DCCOOP provides fairness to all the jobs but DOPTIM does not. DOPTIM provides the lowest expected price for the entire system. Similar behavior of the various schemes is observed for any system utilization when the overhead is 5%.

For overheads above 5%, it was observed that DOPTIM and DCCOOP show superior performance for up to 60% - 70% system utilizations and approach the static schemes for high utilizations. This is because, as the overheads increase, the cost for exchanging the system state information by the dynamic schemes increases which results in a decrease in the performance.

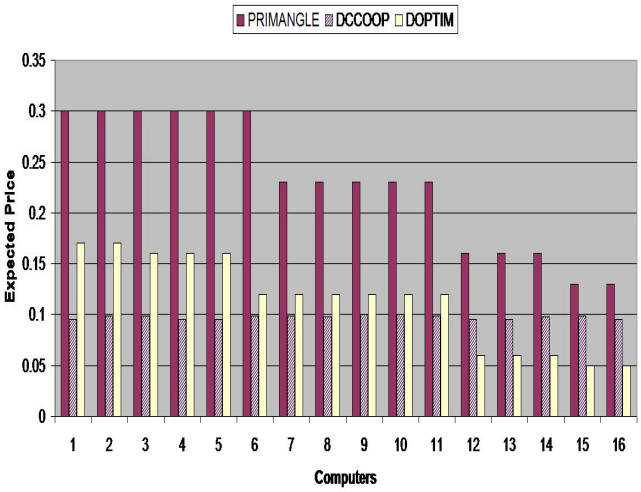

Fig. 5. Expected Price at Computers (OV = 5%)

B. Effect of Bias (Δ)

Δ ($\Delta > 0$) is a *bias* used to protect the system from instability by forbidding the job allocation scheme to react to small load distinctions between the nodes. Figure 6 presents the variation of expected price with system utilization of DCCOOP for various biases. The overhead for job transfer is assumed to be 5%. The other parameters are fixed as in Figure 2. It can be observed that as the bias increases, the expected price of DCCOOP increases. For a high bias (e.g. Δ = 1), the expected price of DCCOOP approaches that of PRIMANGLE. This is because, for a high bias, jobs will be processed locally without being transferred to lightly loaded nodes. Thus the expected response time of the jobs increases which will result in an increase in the expected price to process the jobs. The effect of bias on DOPTIM is similar.

Simulations were also performed to study the effect of the exchange period of state information (P) on DCCOOP and DOPTIM. It was observed that except for large P (e.g. P $>$ 2 secs), the dynamic schemes show superior performance compared to the static schemes. Large P will result in exchanging outdated state information among the nodes and so the performance of the dynamic schemes decreases.

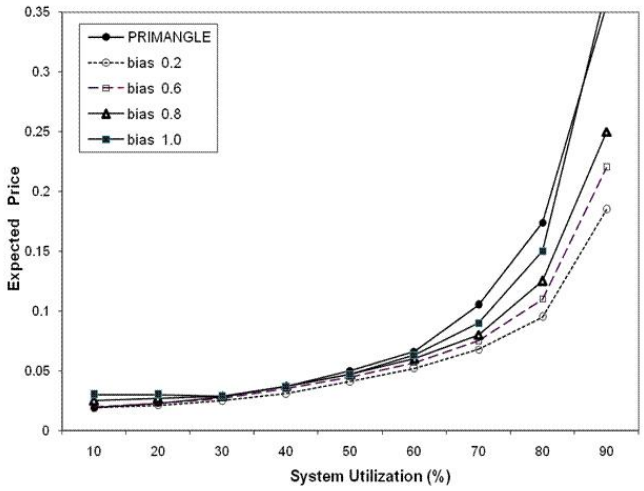

Fig. 6. Effect of Bias

VI. Conclusions

In this paper, two price-based dynamic job allocation schemes for computational grids are proposed. DOPTIM tries to minimize the expected price for the execution of all the jobs in the grid system while DCCOOP tries to provide a fair solution. Simulations are made comparing the proposed dynamic schemes with static job allocation schemes. It was observed that both DOPTIM and DCCOOP show superior performance compared to the static schemes. DOPTIM is advantageous when a system-optimum is desired (*e.g.* when all the jobs belong to the same user or social group). DCCOOP is advantageous when fairness is as important as other performance characteristics (*e.g.* when jobs belong to different users and each user prefers to pay the same unit price as others).

Acknowledgements

Some of the reviewers comments which helped improve the quality of the paper are gratefully acknowledged. This research was funded in part by a grant from the Southern Arkansas University Research Committee.

References

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

[2] N. G. Shivaratri, P. Krueger, and M. Singhal, "Load distributing for locally distributed systems," *IEEE Computer Magazine*, vol. 8, no. 12, pp. 33–44, December 1992.

[3] J. Grossklags and C. Schmidt, "Software agents and market (in) efficiency: a human trader experiment," *IEEE Trans. on Systems, Man and Cybernetics - Part C*, vol. 36, no. 1, pp. 56–67, 2006.

[4] P. Ghosh, N. Roy, S. K. Das, and K. Basu, "A pricing strategy for job allocation in mobile grids using a non-cooperative bargaining theory framework," *Journal of Parallel and Distributed Computing*, vol. 65, no. 11, pp. 1366–1383, 2005.

[5] *Amazon Elastic Compute Cloud*. http://www.amazon.com/.

[6] *On-demand Computing Using network.com*. White paper, Sun Microsystems, Inc., Aug. 2007.

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

[8] 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*, August 2000, pp. 373–382.

[9] Y. Zhang, H. Kameda, and S. L. Hung, "Comparison of dynamic and static load-balancing strategies in heterogeneous distributed systems," *IEE Proc. Computers and Digital Techniques*, vol. 144, no. 2, pp. 100–106, March 1997.

[10] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1022–1034, Sep. 2005.

[11] S. Penmatsa and A. T. Chronopoulos, "Dynamic multi-user load balancing in distributed systems," in *Proc. of the 21st IEEE Intl. Parallel and Distributed Processing Symposium*, Long Beach, California, March 26-30, 2007.

[12] C. Kim and H. Kameda, "An algorithm for optimal static load balancing in distributed computer systems," *IEEE Trans. on Computers*, vol. 41, no. 3, pp. 381–384, March 1992.

[13] L. Anand, D. Ghose, and V. Mani, "Elisa: An estimated load information scheduling algorithm for distributed computing systems," *Computers and Mathematics with Applications*, vol. 37, pp. 57–85, 1999.

[14] Z. Zeng and B. Veeravalli, "Design and performance evaluation of queue-and-rate-adjustment dynamic load balancing policies for distributed networks," *IEEE Transactions on Computers*, vol. 55, no. 11, pp. 1410–1422, Nov 2006.

[15] 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*, May 2002.

[16] C. S. Yeo and R. Buyya, "Pricing for utility-driven resource management and allocation in clusters," *Intl. Jrnl. of High Performance Computing Applications*, vol. 21, no. 4, pp. 405–418, 2007.

[17] Y. Zhang, C. Koelbel, and K. Kennedy, "Relative performance of scheduling algorithms in grid environments," in *7th IEEE Intl. Symp. on Cluster Computing and the Grid*, Brazil, May 2007, pp. 521–528.

[18] L. Xiao, Y. Zhu, L. Ni, and Z. Xu, "Incentive-based scheduling for market-like computational grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 903–913, July 2008.

[19] R. Subrata, A. Zomaya, and B. Landfeldt, "Game theoretic approach for load balancing in computational grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 1, pp. 66–76, Jan. 2008.

[20] Y. K. Kwok, K. Hwang, and S. Song, "Selfish grids: Game-theoretic modeling and NAS/PSA benchmark evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 5, pp. 621–636, May 2007.

[21] K. Rzadca, D. Trystram, and A. Wierzbicki, "Fair game-theoretic resource management in dedicated grids," in *Proc. of the 7th IEEE Intl. Symp. on Cluster Computing and the Grid*, Brazil, May, 2007, pp. 343–350.

[22] S. Penmatsa and A. T. Chronopoulos, "Price-based user-optimal job allocation scheme for grid systems," in *Proc. of the 20th IEEE Intl. Parallel and Distributed Proc. Symp.*, Rhodes Island, Greece, April 25-29, 2006.

[23] R. Shah, B. Veeravalli, and M. Misra, "On the design of adaptive and decentralized load balancing algorithms with load estimation for computational grid environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1675–1686, Dec. 2007.

[24] S. Penmatsa and V. P. Mantena, "Dynamic cost minimization for multi-class jobs in computational grids," in *Proc. of the 24th Intl. Conf. on Computers And Their Applications*, New Orleans, LA, April 8-10, 2009.

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

[26] S. Penmatsa and A. T. Chronopoulos, "Cooperative load balancing for a network of heterogeneous computers," in *Proc. of the 20th IEEE Intl. Parallel and Distributed Proc. Symp.*, Rhodes Island, Greece, April 2006.

[27] R. Cubert and P. Fishwick, "Sim++ reference manual," in *Department of Computer and Information Science and Engineering*, University of Florida, July 1995.

[28] Y. C. Chow and W. H. Kohler, "Models for dynamic load balancing in a heterogeneous multiple processor system," *IEEE Trans. on Computers*, vol. C-28, no. 5, pp. 354–361, May 1979.

[29] P. Ghosh, K. Basu, and S. K. Das, "A game theory-based pricing strategy to support single/multiclass job allocation schemes for bandwidth-constrained distributed computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 289–306, March 2007.