Competitive Equilibrium Approach for Load Balancing a Data Grid

K. Shahu Chatrapati[1], J. Ujwala Rekha[2] and A. Vinaya Babu[3]

[1] **Dept of C.S.E, JNTUH CEJ**
Jagitial, Andhra Pradesh, INDIA

[2] **Dept of C.S.E, JNTUH CEH**
Hyderabad, Andhra Pradesh, INDIA

[3] **Dept of C.S.E, JNTUH CEH**
Hyderabad, Andhra Pradesh, INDIA

Abstract

Distributed data-intensive applications generate a large number of tasks/jobs, that need for its execution two are more data sets, that are replicated and scattered on various storage repositories that are connected to each other, and computational sites through networks of varying capability. To get the best performance, load balancing strategies for Data Grids, should judiciously select the dataset replicas to be used and the site where these will accumulate for the job to be executed. This paper studies *Global Optimal Scheme* (GOS), *Nash Equilibrium Scheme* (NES) and then proposes pricing mechanism using *Competitive Equilibrium Approach*. A computer model is run to evaluate the performance of these techniques. The results show that GOS minimizes mean response time of the entire set of jobs, without taking into account response time of each job individually, and NES minimizes the response time of each job individually, without taking into account mean response time of all jobs. The proposed *Competitive Equilibrium Scheme* (CES) simultaneously minimizes mean response time of all jobs, and the response time of each job individually. Another performance metric considered is the network load imposed by jobs in transferring datasets from storage repositories to the computational sites. The results show that the network load is the least in CES.

***Keywords**: Load Balancing, Data Grid, Nash Equilibrium, Competitive Equilibrium, Data-intensive application*

1. Introduction

Distributed data intensive applications generate a large number of tasks/jobs that reference datasets may be of the order of Giga bytes and higher, like for example in the high energy physics [20] or life sciences [4] applications. These datasets may be each replicated on various storage repositories that are connected to each other and to the computational sites through networks of varying capability. Thus a job initiated at one site can be executed at any of the computational sites, and access the datasets it requires from any of the subset of storage repositories, such that each dataset required for the job can be obtained from one of the storage repositories in the set. Since the data sets are very large, transferring them from the storage repositories to the eventual point of execution produces a noticeable impact on the response time of the job, as well as the network load. Load balancing strategies for Data Grids therefore, should judiciously select a computational resource and a set of storage resources for each job, so as to minimize the response time of each job, as well as the network load.

In this study we address the problem of load balancing as consisting of two parts. In the first part, called *data management* the data set replicas that the job will use are selected. In the second part, called *job allocation*, the site where these data sets will consolidate for the job's execution is decided. In this paper, we study two approaches for load balancing. *Global Optimal Scheme* and *Nash Equilibrium Scheme* and then propose a pricing mechanism using *Competitive Equilibrium Approach*. The Global Optimal Scheme as in [5], tries to minimize the response time of the entire set of jobs. The Nash Equilibrium scheme as in [1], tries to minimize the response time of each job individually. The proposed *Competitive Equilibrium Scheme* tries to minimize the mean response time of the entire set of jobs, as well as the individual response time of each job simultaneously.

The remainder of the paper is organized as follows. In section 2 we report on previous work. In section 3, we present an overview of the proposed model including the grid model and the application model, and the objective function. Sections 4 and 5 present load balancing using Global Optimal Scheme and Nash Equilibrium Scheme respectively. In section 6 we discuss Competitive Equilibrium theory and describe how it can be applied to load balancing a data grid. Section 7 presents the simulation environment and performance results obtained for three schemes. Finally conclusions are presented in section 8.

2. Previous Work

Grid Networks can be distinguished as either *computational* or *data grid* depending on whether they serve *computationally intensive* or *data-intensive applications*. A large body of work has been done in the area of load balancing a computational grid [18][19][21], where *centralized*, *hierarchical* and *distributed scheduling* schemes are presented. Performance metrics used in such schemes are *average task delay*, *average slow down* and several others. *Fair scheduling* in grid networks has also been addressed in [10][15][13].

Data management in grids can be divided into four camps namely, (1) *Architecture and management* for data replication [2], (2) *Data replication placement*, (3) *Data replication selection* and (4) *Data consistency*. *Data replication* is the process of distributing replicas of data sets across the sites. Data replication reduces access time and bandwidth consumption and improves reliability and availability [1][6]. When different sites hold replicas of particular data set, selecting the best replica among them optimizes desired performance criterion such as the access time, network load [16][17] and is called *data replication selection*.

There are a handful of works that jointly addressed the *task scheduling* and *data replication* problems [12][8][3]. They either minimized mean response time of all jobs or response time of each job individually but not both.

In this work we jointly address two problems namely, task scheduling/job allocation and data replica selection. This can be applied to data intensive applications that need for their execution more than one pieces of data replicated at different sites. Contrary to the above mentioned prior work, we propose competitive equilibrium approach, a pricing mechanism that simultaneously minimizes mean response time of all jobs and mean response time of each job individually.

3. Proposed Work

In this section we describe the grid system and application models considered in our work and later formulate the objective function for load balancing the grid system.

3.1 Grid System model

We considered a grid system consisting of a set of $\mathbf{n}$ computing resources/sites $\mathbf{R}=\{\mathbf{r_1},...,\mathbf{r_n}\}$ and a set of $\mathbf{p}$ storage repositories/sites $\mathbf{D}=\{\mathbf{d_1},...,\mathbf{d_n}\}$. A computing site $\mathbf{r_i}$ is described by two constants: (1) its computational capacity $\boldsymbol{\mu}_\mathbf{i}$ measured in computation units per second (e.g, Million Instructions per Second-MIPS) and (2) its storage capacity $\mathbf{w_i}$, measured in data units (e.g., bytes). The available bandwidth between the storage site $\mathbf{d_j}$ to computing site $\mathbf{r_i}$ is $\mathbf{B_{ji}}$.

Data is organized in the form of datasets. A dataset can be an aggregated set of files, a set of records, or even a part of a large file. Datasets are replicated on the storage sites by a separate replication process, which considers various factors such as locality of access, load on the storage repositories, and available storage space. We assume that the replication is already done. Hence replication process and storage capacity of a storage repository are of no interest to us but the focus will be on dataset replica selection.

3.2 Application model

The application is composed of a set of $\mathbf{m}$ jobs $\mathbf{J}=\{\mathbf{j_1},...,\mathbf{j_m}\}$ without interdependencies and accesses a set of $\mathbf{l}$ datasets $\mathbf{I}=\{\mathbf{i_1},...,\mathbf{i_l}\}$, which are distributed on members of $\mathbf{D}$. The size of dataset $\mathbf{i_a}$ is $\mathbf{size(i_a)}$. Also for a dataset $\mathbf{i_a}\in\mathbf{I}$, $\mathbf{D_{i_a}}\subseteq\mathbf{D}$ is the set of storage sites on which $\mathbf{i_a}$ is replicated. Also $\mathbf{D_{i_b}}$ and $\mathbf{D_{i_c}}$ need not be pair wise disjoint for every $\mathbf{i_b}$, $\mathbf{i_c}\in\mathbf{I}$. A job $\mathbf{j_k}\in\mathbf{J}$ processes a subset of $\mathbf{I}$ denoted by $\mathbf{I_k}$ where $\mathbf{I_k}\subseteq\mathbf{I}$, and contains $\mathbf{l_k}$ number of datasets. The amount of computation needed to execute job $\mathbf{j_k}$ is $\boldsymbol{\alpha}*\mathbf{size(I_k)}$, where $\mathbf{size(I_k)}$ is the total size of all datasets in $\mathbf{I_k}$ given by $\mathbf{size(I_k)}=\sum_{\mathbf{i_a}\in\mathbf{I_k}}\mathbf{size(i_a)}$ and the value $\boldsymbol{\alpha}$ can take from 0 to 1. The parameter $\boldsymbol{\alpha}$ defines a trade off between computation and data-insensitivity of jobs. As the value of **α** decreases the jobs have less computation demands.

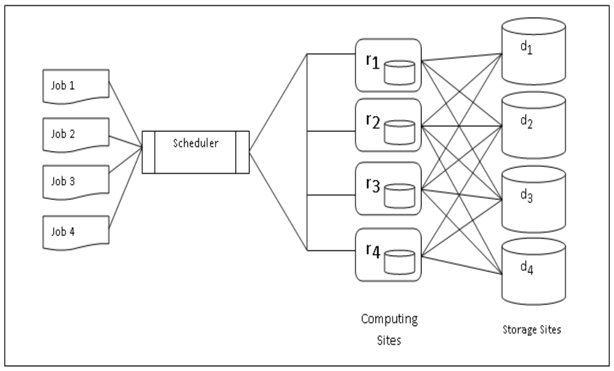

Figure 1. Relationship of jobs, scheduler, computing sites, and storage sites

All the jobs generated by the application are submitted to the scheduler. For each job $\mathbf{j_k}$, the scheduler should choose one of the computing resources for executing the job and one storage repository each for accessing each of $\mathbf{l_k}$ datasets required by the job. The computing resource and the storage repositories thus selected are collectively referred to as the resource set, associated with job $\mathbf{j_k}$ and is denoted by $\mathbf{S^k} = \{\mathbf{R^k},\mathbf{D^k}\}$ where $\mathbf{R^k} \subseteq \mathbf{R}$ is a singleton representing the computing resource selected for executing the job $\mathbf{j_k}$, and $\mathbf{D^k}$ is an $\mathbf{l'_k}$ sized set of storage sites chosen for accessing the datasets required by the job $\mathbf{j_k}$. Since multiple datasets can be retrieved from a single data host $\mathbf{l'_k} \le \mathbf{l_k}$. Figure 1 shows the relationship among jobs, scheduler, computing sites and storage sites.

3.3 Objective Function

The response time of a job in a system defined as above consists of node delay (queuing delay + processing delay) at computing site and also communication delay (queuing delay + transfer time) incurred due to the transfer of datasets from storage sites to the computing site.

Let job $\mathbf{j_k}$ be scheduled to be executed at computing site $\mathbf{r_i}$ (for some value of i=1,..,n), then $\mathbf{x_{ki}} = 1$ and $\mathbf{x_{ki'}} = 0$ for all values of $i' \neq i$. A set of values for $\mathbf{x_{ki}}$ (k=1,…,m; i=1,…,n) are to be chosen by the scheduler taking into account node delay (queuing delay + processing delay), and storage space available, at each computing site. That is, the size of datasets accessed by various jobs scheduled to be executed at computing site $\mathbf{r_i}$, should not exceed its storage capacity $\mathbf{w_i}$. In other words,

$$\sum_{k=1}^{m} \mathbf{x_{ki}} * \mathbf{size(I_k)} \le \mathbf{w_i} \qquad \text{for all } i = 1,...,n \quad (1)$$

For each job $\mathbf{j_k}$ $(k = 1,..m)$ scheduled to be executed at one of the computing sites $\mathbf{r_i}$ (for some values of i = 1,..,n) and for each dataset $\mathbf{i_a} \in \mathbf{I_k}$, the scheduler should choose one of the storage sites $\mathbf{d_j} \in \mathbf{D_{i_a}}$ for retrieval, taking into account communication delay (queuing delay + transfer delay) in retrieving dataset $\mathbf{i_a}$ from storage site $\mathbf{d_j}$ to computing site $\mathbf{r_i}$. For job $\mathbf{j_k} \in \mathbf{J}$, let $\mathbf{i_a} \in \mathbf{I_k}$ be selected from storage site $\mathbf{d_j} \in \mathbf{D_{i_a}}$ for retrieval, then $\mathbf{y^k_{ja}} = 1$ and $\mathbf{y^k_{j'a}} = 0$ for all values of $\mathbf{j'} \neq \mathbf{j}$.

The average size of data processed by any job that is scheduled to be executed at computing site $\mathbf{r_i}$ can be computed as

$$JS(\mathbf{r_i}) = \frac{1}{\mathbf{m_i}} * \sum_{k=1}^{m} \mathbf{size(I_k)} * \mathbf{x_{ki}} \qquad (2)$$

where $\mathbf{m_i}$ is the number of jobs scheduled to be executed at computing site $\mathbf{r_i}$. The average time to process any job $\mathbf{j_k}$ scheduled to be executed at computing site $\mathbf{r_i}$ be calculated as

$$PD(\mathbf{j_k},\mathbf{r_i}) = \frac{\alpha .* \sum_{k'=1}^{m} (\mathbf{size(I_{k'})} * \mathbf{x_{k'i}})}{\mathbf{\mu_i} * \mathbf{m_i}} \qquad (3)$$

The average node delay (queuing delay + processing delay) of any job $\mathbf{j_k}$ scheduled to be executed at computing site $\mathbf{r_i}$ is therefore given as

$$ND(j_k, r_i) = PD(j_k, r_i) * \frac{(1+2+....+m_i)}{m_i}$$
$$= \frac{\alpha * \sum_{k'=1}^{m} (size(I_{k'i}) * x_{k'i})}{\mu_i * m_i} * \frac{m_i * (m_i + 1)}{2 * m_i}$$
$$\approx \frac{\alpha * \sum_{k'=1}^{m} (size(I_{k'i}) * x_{k'i})}{2 * \mu_i} \qquad (4)$$

The average size of datasets requested from storage site d_j to computing site r_i can be computed as

$$DS(d_j, r_i) = \frac{1}{l_{ji}} \sum_{k=1}^{m} \sum_{a=1}^{l} (y_{ja}^k * size(i_a) * x_{ki}) \qquad (5)$$

where l_{ji} is the number of datasets requested from storage site d_j to computing site r_i. The average time for transferring any dataset from storage site d_j to computing site r_i is

$$TD(d_j, r_i) = \frac{\sum_{k=1}^{m} \sum_{a=1}^{l} (y_{ja}^k * size(i_a) * x_{ki})}{l_{ji} * B_{ji}} \qquad (6)$$

Therefore the average communication delay (queuing delay+ transfer time) for transferring a dataset i_a, from storage site $d_j \in D_{i_a}$ to computing site r_i is given as

$$CD(i_a, d_j, r_i) = TD(d_j, r_i) * \frac{(1+2+...+l_{ji})}{l_{ji}}$$
$$= \frac{\sum_{k=1}^{m} \sum_{a'=1}^{l} (y_{ja'}^k * size(i_{a'}) * x_{ki})}{l_{ji} * B_{ji}} * \frac{l_{ji} * (l_{ji} + 1)}{2 * l_{ji}}$$
$$\approx \frac{\sum_{k=1}^{m} \sum_{a'=1}^{l} (y_{ja'}^k * size(i_{a'}) * x_{ki})}{2 * B_{ji}} \qquad (7)$$

Let job j_k be scheduled to be executed at r_i, and let each $i_a \in I_k$ be selected from one of the storage sites $d_j \in D_{i_a}$ for retrieval. Then the resource set of job j_k is $S^k = \{R^k, D^k\}$, where $R^k = \{r_i / x_{ki} = 1\}$ and $D^k = \{d_j / y_{ja}^k = 1, \ \forall \ i_a \in I_k\}$. The expected response time of job j_k consists of node delay at computing site r_i for execution, and communication delay for retrieving the datasets $i_a \in I_k$ from selected storage sites

$$RT(j_k, R^k, D^k) = ND(j_k, r_i) + \max_{i_a \in I_k} \{CD(i_a, d_j, r_i)\} \qquad (8)$$

where $d_j \in D^k$ is the storage site such that $y_{ja}^k = 1$, and $r_i \in R$ is the computing site such that $x_{ki} = 1$.
Also, the expected mean response time of the entire set of jobs is

$$MRT = \frac{1}{m} * \sum_{k=1}^{m} \sum_{i=1}^{n} x_{ki} * RT(j_k, R^k, D^k) \qquad (9)$$

where, for each job j_k, $r_i \in R^k$ is the computing site such that $x_{ki} = 1$ and $d_j \in D^k$ is the storage site from where dataset i_a is retrieved such that $y_{ja}^k = 1$.

4. Global Optimal Scheme

This scheme minimizes the mean response time of the entire set of jobs. In this scheme the values for x_{ki} (k=1,..,m; i=1,..,n) and y_{ja}^k k=1,..,m; j=1,…p; a=1,..,l) are obtained by solving the following optimization problem

$$\min_{R^k, D^k} MRT \qquad (10)$$

subject to the constraints given below
1. for all values of k=1,…,m
   (i) if $x_{ki} = 1$ (for some $i = 1,...,n$),
   then $x_{ki'} = 0$ (for $\forall \, i' \neq i$) (11)
   (ii) if $y_{ja}^k = 1$ (for some $i_a \in I_k$ and $d_j \in D_{i_a}$),
   then $y_{j'a}^k = 0$ (for all values of $j' \neq j$) and
   $y_{ja'}^k = 0$ (for all $i_a \notin I_k$ and $j = 1,...,p$) (12)
2. $\sum_{k=1}^{m} x_{ki} * size(I_k) \leq w_i$ (for all i=1,...,n) (13)

We present below the algorithm for computing solution for GOS.

4.1 Algorithm (GOS)

```
Input
      • Computational capacities of
        computing sites: $\mu_1,...,\mu_n$
```

```
    • Storage capacities of computing sites: $w_1,\ldots,w_n$
    • Size of each dataset: $size(i_1),\ldots,size(i_l)$
    • Replica sites of each dataset: $D_{i_a},\ldots,D_{i_l}$
    • Datasets processed by each job: $I_1,\ldots,I_{l_k}$
Output
    • Job allocations:
      $x_{ki}$ for all k=1,…,m and i=1,…,n
    • Replicas:
      $y^k_{ja}$ for all k=1,…,m; j=1,…,p; and a=1,…,l

1.  Initialization
 1.1.  $x'_{ki} = 0$ for all k=1,…,m and i=1,…,n
 1.2.  $y'^k_{ja} = 0$ for all k=1,…,m; j=1,…,p; and a=1,…,l
 1.3.  MRT=Maximum-Value
2.  Repeat steps 2.1. to 2.3.
    Until there is no change
 2.1.  For each job $j_k$ k=1,…,m do
   2.1.1.  $x'_{ki} = 0$ for all i=1,…,n
   2.1.2.  $y'^k_{ja} = 0$ for all k=1,…,m;j=1,…,p; a=1,…,l;
   2.1.3.  Choose the computing site $r_s$, where the storage space is
           available for job $j_k$, and the average node delay (determined
           from (4))is minimum and set $x'_{ks} = 1$
   2.1.4.  For each dataset $i_a \in I_k$ do
      2.1.4.1.  Choose $d_t \in D_{i_a}$ such that the communication delay
                incurred by moving $i_a$ from $d_t$ to $r_s$ (determined
                from (7)) is minimum, and set $y'^k_{ta} = 1$
 2.2.  Compute MRT′ from (9)
 2.3.  IF MRT′ is better than MRT, then
       $x_{ki} = x'_{ki}$ for all k=1,…,m and
       MRT=MRT′
```

5. Nash Equilibrium Scheme

In this section, we formulate the load balancing problem as a non-cooperative game among the jobs. One of the corners stones of non-cooperative game theory is the notion of Nash equilibrium [9]. Nash equilibrium is a choice of strategies by the players where each player's strategies are a best response to the other player's strategies. In other words, if we consider that each player's goal is to minimize his objective function, then no player can decrease the value of his objective function by unilaterally deviating from the equilibrium.
A non-cooperative load balancing game consists of a set of players, a set of strategies and preferences over the set of strategy profiles, where

(i) Players are jobs
(ii) Strategies of each job are the set of feasible job allocation strategies and the set of replica selection strategies for all datasets accessed by the job.
(iii) Preference of each job is represented by the expected response time $RT(j_k, R^k, D^k)$.

Therefore, the goal of each job j_k is to find a job allocation strategy x_{ki} (for all i=1,…,n) and replica selection strategy y^k_{ja} (for all j=1,…,p; a=1,…,l) such that its expected response time is minimized independently of other jobs. The best response for job j_k is a solution to the following optimization problem

$$\min_{r_i \in R} \left\{ ND(j_k, r_i) + \max_{i_a \in I_k} \left\{ \min_{d_j \in D_{i_a}} \left\{ CD(i_a, d_j, r_i) \right\} \right\} \right\} \quad (14)$$

subject to the constraints (11), (12), and (13).

Nash equilibrium solution is obtained by first initializing x_{ki} and y^k_{ja} to zero values (for all k=1,…,m; i=1,…,n; j=1,…,p; a=1,…,l).Then each job updates its strategy by solving the optimization problem (14) one after other. Nash equilibrium is reached when no player can change its strategy and decrease its response time by choosing a different strategy when other user's strategies are fixed.

5.1 Algorithm (NES)

```
Input
    • Computational capacities of computing sites: $\mu_1,\ldots,\mu_n$
    • Storage capacities of computing sites : $w_1,\ldots,w_n$
    • Size of each dataset: $size(i_1),\ldots,size(i_l)$
    • Replica sites of each dataset: $D_{i_a},\ldots,D_{i_l}$
    • Datasets processed by each job: $I_1,\ldots,I_{l_k}$
Output
    • Job allocations :
        $x_{ki}$ for all $k=1,\ldots,m$ and $i=1,\ldots,n$
    • Replicas:
        $y^k_{ja}$ for all k=1,…,m; j=1,…,p; and a=1,…,l
1.Initialization
  1.1 $x_{ki}=0$ for all $k=1,\ldots,m$ and $i=1,\ldots,n$
  1.2 $y'^k_{ja}=0$ for all k=1,…,m; j=1,…,p; and a=1,…,l
2. Repeat step 2.1.(Until there is no change)
  2.1. For each job $j_k$ k=1,…,m do
    2.1.1 $x_{ki}=0$ for all i=1,…,n.
    2.1.2 $y^k_{ja}=0$ for all k=1,…,m; j=1,…,p; a=1,…,l;
    2.1.3 Choose the computing site $r_s$ ,where the storage space is
          available for job $j_k$ ,and the average node delay (determined
          from (4))is minimum and set $x_{ks}=1$
    2.1.4 For each dataset $i_a \in I_k$ do
      2.1.4.1. Choose $d_t \in D_{i_a}$ such that the communication delay incurred
               by moving $i_a$ from $d_t$ to $r_s$ (determined from (7)) is
               minimum, and set $y^k_{ta}=1$.
    2.1.5. Compute $RT(j_k, R^k, D^k)$ .
```

6. Proposed solution

In this section we discuss competitive equilibrium theory, in which two market models namely, Walrasian and Fisher's market model are described. Later in this section, the load balancing problem is translated to Fisher's market model, and an algorithm for computing competitive equilibrium is given.

6.1 Competitive Equilibrium Theory

In the capitalist economy, crucial regulatory functions such as ensuring stability, efficiency, and fairness are relegated to pricing mechanisms. Thus, competitive equilibrium theory of equilibrium prices gained an important place in mathematical economics. The theory dates back to 1870s and the credit for initiating the study can be attributing to French Economist Lêon Walras [14]. In Walrasian model the market consists of **m** agents and **n** divisible goods. Let b_{ij} and c_{ij} denote respectively, the non-negative endowment and consumption by agent *i*, relative to the good *j*. Let p_j denote the non-negative price associated to the good *j*. Grouping the introduced quantities into vectors, the total endowment vector of agent *i* is $b_i=(b_{i1},\ldots,b_{in})$, the total consumption vector of agent *i* is $c_i=(c_{i1},\ldots,c_{in})$ and the price vector $P=(p_1,\ldots,p_n)$. Let $u_i(c_i): R^n_+ \to R_+$, describe the preference of agent *i* for different bundles of goods. At given prices of goods, each agent sells their initial endowment, and buys a bundle of goods, which maximizes $u_i(c_i)$ subject to her budget constraints.

An equilibrium is a set of market clearing prices $P=(p_1,\ldots,p_n)$, such that for agent *i*, there is a bundle of goods $c_i=(c_{i1},\ldots,c_{in})$ such that the following two conditions hold:

i. For each agent *i*, the vector c_i maximizes $u_i(c_i)$ subject to the constraints

$$\sum_{k=1}^{n} p_k * c_{ik} \le \sum_{k=1}^{n} p_k * b_{ik} \quad (15)$$

ii. For each good *j*,

$$\sum_{k=1}^{m} c_{kj} \le \sum_{k=1}^{m} b_{kj} \quad (16)$$

According to Arrow and Debreu Theorem [11], such an equilibrium exists under very mild conditions, if the utility functions are concave. The first fundamental Theorem asserts that such equilibrium is pareto efficient [11].

Fisher [7] in 1891 independently modeled a market, which consists of a set ***m*** buyers and a set of ***n*** divisible goods. Each buyer *i* has an amount $\mathbf{e_i}$ of money and each good *j* has an amount $\mathbf{f_j}$ .of this good. The preferences of agent *i* for different bundles of goods is denoted by $u_i(\mathbf{c_i}): R_+^n \to R$, where $\mathbf{c_i} = (\mathbf{c_{i1}}, ..., \mathbf{c_{in}})$ is the consumption vector of agent *i*, and $\mathbf{c_{ij}}$ is the consumption of good *j* by agent *i*. Equilibrium prices is an assignment of prices of $\mathbf{P} = (\mathbf{p_1}, ..., \mathbf{p_n})$ to the goods, such that the market clears i.e., there is neither shortage nor surplus. In other words, the following two conditions should hold:

i. For each buyer *i*, the vector $\mathbf{c_i}$ maximizes $u_i(\mathbf{c_i})$ subject to the constraints

$$\sum_{k=1}^{n} \mathbf{p_k} * \mathbf{c_{ik}} \le \mathbf{e_i} \qquad (17)$$

ii. For each good *j*,

$$\sum_{k=1}^{m} \mathbf{c_{kj}} \le \mathbf{f_j} \qquad (18)$$

In this study, load balancing problem is translated to Fisher's market model, where buyers are jobs, and goods are of different kinds, namely computing resources, and communication links between storage sites to computing sites.

6.2. Competitive Equilibrium Approach for Load Balancing

At first, the proposed model described in section 3 is translated to Fisher's market model, where buyers are jobs, and goods are computing resources, and the communication links between storage sites and computing sites. Each job $\mathbf{j_k}$ is endowed a monetary budget $\mathbf{b_k} \ge 0$ $\mathbf{B} = (\mathbf{b_1}, ..., \mathbf{b_m})$ to purchase the computing power and bandwidth, and has utility function $U(\mathbf{j_k}, \mathbf{R^k}, \mathbf{D^K}) = -RT(\mathbf{j_k}, \mathbf{R^k}, \mathbf{D^K})$ to denote her preferences for various computing resources and communication links. The price for executing unit computation at computing resource $\mathbf{r_i}$ is $\mathbf{p_i}$ and the price for transferring a unit size of data through the communication link from storage site $\mathbf{d_j}$ to computing site $\mathbf{r_i}$ is $\mathbf{q_{ji}}$.

Let job $\mathbf{j_k}$ be scheduled to be executed at computing site $\mathbf{r_i} \in \mathbf{R}$, and for each dataset $\mathbf{i_a} \in \mathbf{I_k}$ that job $\mathbf{j_k}$ processes, let storage site $\mathbf{d_a} \in \mathbf{D_{i_a}}$ be selected for retrieval, then the cost incurred by job $\mathbf{j_k}$, can be calculated as follows

$$\mathbf{c_k} = \mathbf{p_i} * \alpha * \mathbf{size(I_k)} + \sum_{\mathbf{i_a} \in \mathbf{I_k}} \mathbf{q_{ai}} * \mathbf{size(i_a)} \qquad (19)$$

The competitive equilibrium solution to load balancing constitutes- (i) finding a set of prices $\mathbf{P} = \{\mathbf{p_1}, ..., \mathbf{p_n}, \mathbf{q_{11}}, ..., \mathbf{q_{1n}}, ..., \mathbf{q_{p1}}, ..., \mathbf{q_{pn}}\}$ to computing resources and communication links, (ii) At prices **P** allocation of jobs to computing resources, and for each job, selection of dataset replicas for the datasets it processes, is such that each job maximizes her utility $U(\mathbf{j_k}, \mathbf{R^k}, \mathbf{D^K})$ subject to her budget constraints and the market clears i.e,

$$\max_{\mathbf{R^k}, \mathbf{D^k}} \{U(\mathbf{j_k}, \mathbf{R^k}, \mathbf{D^K})\} \text{ for all } k=1,...,n \qquad (20)$$

subject to the constraints (11) to (13) and the market clearing condition given by

$$\mathbf{c_k} \le \mathbf{b_k} \quad \text{for all } k=1,...,m \qquad (21)$$

This is an artificial exchange of money where the price **P** and budget **B** are not real money and do not have any physical interpretations. The meaningful output is only load distribution $\mathbf{x_{ki}}$ (for all k=1,…,m; i=1….,n) and dataset replica selection for each job $\mathbf{y_{ja}^k}$ (for all k=1,…,m; j=1,…,p; a=1,…,l). The budget **B** and price **P** have no outside use; they are only an economic means for defining user's strategy profile to achieve individual and system optimality.

Walras [14] introduced a price-adjustment process called tâtonment trial and error process run by a fictitious auctioneers. The buyers take the prices as given, and report their demands at these prices to auctioneer. The auctioneer, then adjusts the prices in proportion to the magnitude of the aggregate demands, and announces the new prices. In each iteration, the buyer recalculates their demands upon receiving the newly adjusted prices and reports these new demands to the auctioneer. The process continues until prices converge to equilibrium.

We present below the algorithm for obtaining solution to the competitive equilibrium scheme.

6.3 Algorithm (CES)

```
Input
  • Computational capacities of computing sites: $\mu_1,...,\mu_n$
  • Storage capacities of computing sites: $w_1,...,w_n$
  • Size of each dataset : $size(i_1),...,size(i_l)$
  • Replica sites of each dataset: $D_{i_a},...,D_{i_l}$
  • Datasets processed by each job: $I_1,....,I_{l_k}$
Output
  • Job allocations : $x_{ki}$ for all $k=1,…,m$ and $i=1,…,n$
  • Replicas: $y^k_{ja}$ for all $k=1,…,m; j=1,…,p;$ and $a=1,…,l$
1. Initialization
    1.1. $b_k \leftarrow m \quad \forall j=1,...,m$
    1.2. $p_i \leftarrow 1 \quad \forall i=1,...,n$
    1.3. $q_{ji} \leftarrow 1 \quad \forall j=1,...,p; i=1,...,n$
2. Loop through steps (2.1) through (2.3) until ($\beta \le$ error tolerance)
    2.1. At prices P compute $x_{ki}$ (for all k=1,…,m; i=1,…,n) and $y^k_{ja}$ (for all k=1,…,m; j=1,…,p; a=1,…,l) such that each job maximizes her utility (20) subject to the constraints (11) to (13) and (21).
    2.2. Obtain market clearing error
         $\beta = \sqrt{\sum_{k=1}^{m} \xi_k^2}$     (22)
         Where $\xi_k$ is given by
         $\xi_k = b_k - c_k$     (23)
    2.3. Adjust the prices P in proportion to aggregate demands according to law of supply and demand
```

7. Experiments and Results

A computer model is run, in order to evaluate the effectiveness of Competitive Equilibrium Scheme (CES) against Global Optimal Scheme (GOS), and Nash Equilibrium Scheme (NES). The performance metrics used are the mean response time of the entire set of jobs, individual response time of each job, mean communication delay from various storage sites to computing sites, and mean job delay at each site.

The parameters used for the experiments are given below.

i. The grid system consists of 6 computing resources with service rates as shown in Table I. The storage capacities of all computing resources are assumed to be 15GB.

Table I Service Rates of Computing Resources

Computing resources	r_1	r_2	r_3	r_4	r_5	r_6
Service rates (MIPS)	1	2	3	4	5	6

ii. The system consists of 6 storage repositories each of 30 GB capacity. The available bandwidth between storage sites and computing sites is shown in Table II

iii. The application consists of 15 data sets and 10 jobs. The sizes of datasets are in the range of 500MB to 2GB.The jobs are configured to require datasets for their execution ranging from 3 to 8 datasets. We assume that a replica of each dataset is present on all storage sites.

Table II Available Bandwidth between storage & computing sites in Mbps

Storage Repositories	Computing Sites					
	r_1	r_2	r_3	r_4	r_5	r_6
d_1	1	2	3	4	5	6
d_2	2	3	4	5	6	1
d_3	3	4	5	6	1	2
d_4	4	5	6	1	2	3
d_5	5	6	1	2	3	4
d_6	6	1	2	3	4	5

iv. The access pattern follows a power (zipf) distribution as it more realistic and has been used in the literature to model dataset frequency of use. This is to indicate that some datasets would be used more than others according to a power law. The probability of a dataset being used by a job is $p_{i}=\frac{i^{-\delta}}{\gamma}$, where δ defines the shape of the exponential curve. We set δ to 2, in our experiments and γ is normalization constant and is given by $\gamma=\sum_{i=1}^{l} i^{-\delta}$.

Figure 2 illustrates, the mean response time of the entire set of jobs, as the jobs become more CPU- than data-intensive. In order to examine the effect, the value of **α** is varied from 0.01 to 1. We observe that, the mean response time of all jobs increases with CPU-insensitivity. More over, we observe that the mean response time of all jobs is least in GOS, while largest in NES.

Figure 3 presents individual response time of each job. The values of **α** is kept constants at 0.01. We observe that the difference in response times for CES is very small compared to GOS, and NES. We conclude from figures 2 and 3 that CES optimizes mean response time of all jobs, and individual response time of each job simultaneously.

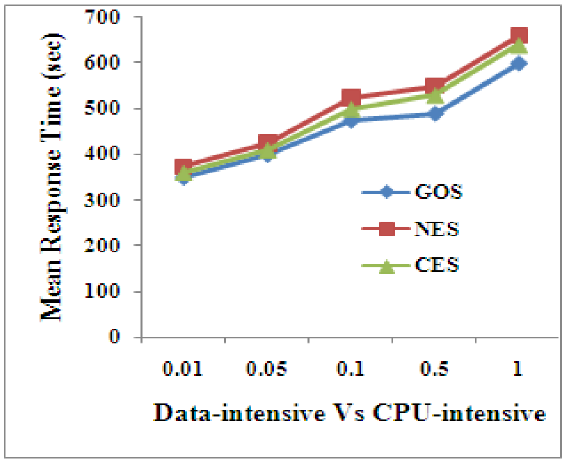

Figure 2. CPU-Intensitivity Vs Mean Response Time of The Entire Set of Jobs

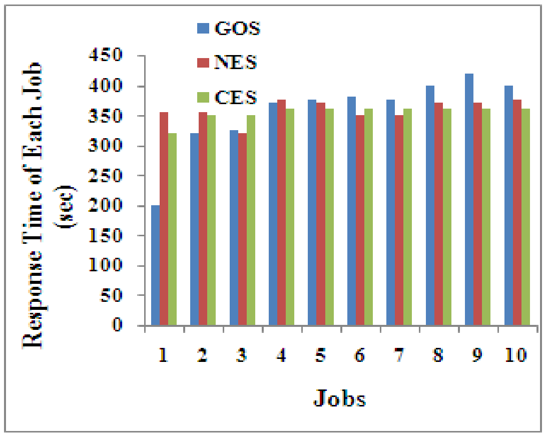

Figure 3. Individual Response Time of Each Job (α=0.01)

Figures 4 and 5 present mean communication delay to each computing site from various storage sites, and mean job delay at each site respectively. The values of **α** is kept constants at 0.01. We observe that the differences in mean communication delays in figure 4, and differences in mean job delays in figure 5 are least in CES, compared to GOS and NES. Also, the communication delay to the computing sites that have higher service rates is large in the case of GOS, i.e., the traffic to those sites is very much. Moreover, the mean job delay is very high on computing sites that have lower service rates in the case of GOS, while it is almost same for every computing site in the case of CES. However the network load on all communication links, and computation load on all computing sites are balanced very much in CES.

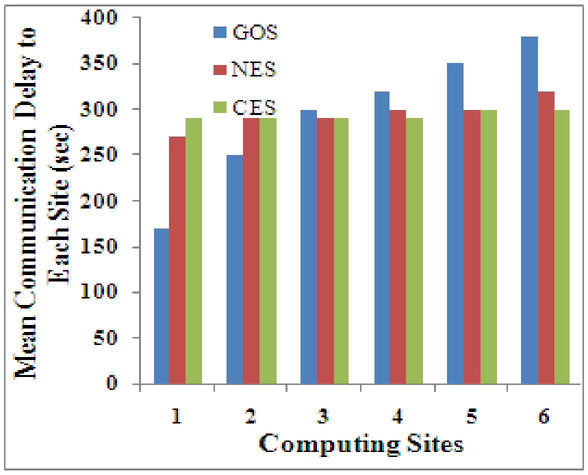

Figure 4. Mean Communication Delay to Each Computing Site (α=0.01)

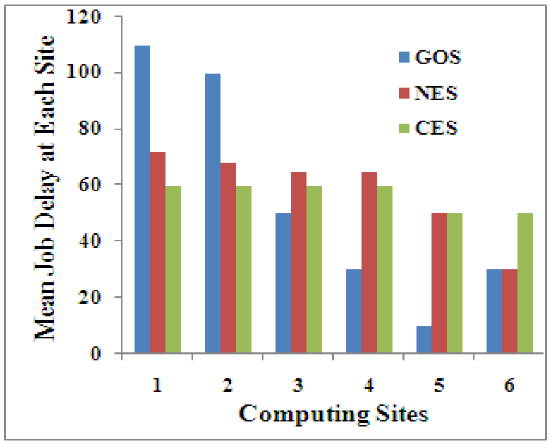

Figure 5. Mean Job Delay at Each Site (α=0.01)

8. Conclusion

In this paper we have addressed the load balancing problem in data grids and proposed a pricing mechanism using competitive equilibrium approach. Our approach directly takes into account the self interest of individual jobs. Prices of the resources are determined in such a way that, when each job competes for maximizing its utility under its budget constraints, the market clears. We show that such a condition on the market has the effect of driving down the mean job delay of the entire set of jobs, as well as individual delays. We also show that as a result network load on all communication links as well as computing load on all computers is balanced.

References

[1] Ali H.Elghirani, Riky Subrata, Albert Y.Zomaya, “A Proactive Game-Theoretic Framework for Data Replication in Data Grids”, In the proceedings of eighth IEEE International Symposium Cluster Computing and the Grid, pp.433-440 (2008)

[2] A.Chervenak, I.Foster, C.Kesselmen, C.Salisbury, S. Tuecke, “The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets”, Journal of Network and computer Applications, 23(2),187-200,July 2000.

[3] A.Elghirani, Riky Subrata, Albert Y.Zomaya, “Intelligent Scheduling and Replication in Data grids: A synergestic Approach”, International Symposium on cluster computing and the Grid, pp.179-182,2007.

[4] A.Krishnan, “A survey of Life Sciences Applications on the Grid”, New Generation computing, 22:111=126,(2004).

[5] D.Grosu, A.T. Chronopoulous, M.Y.Leung, “Cooperative Load Balancing in Distributed Systems”, Concurrency and Computation: Practices and Experience, 20(16), 1953-1976(2008).

[6] H.Lamehamedi, B.Szymanski, Z.Shenter, E. Deelman, “Data Replication Strategies in Grid Environments”, In the proceedings of fifth International conference on Algorithms and Architecture for Parallel Processing, Pages 378-383, (2002).

[7] H.Scarf, The Computations of Economic Equilibria, In cowles Foundation monograph No.24, Yale University Press, New Haven (1973).

[8] H.Shan, L.Oliker, W.Smith, R.Biswas, “Scheduling in Heterogeneous Grid Environment: The effects of Data Migration”, International Conference on Advanced Computing and Communication, 2004.

[9] J.Nash, Non-Cooperative games, Ann. Math., 54(2), 286-295 (1951).

[10] K.H.Kim, R.Buyya,” Fair Resource Sharing in Hierarchical Virtual Organization for Global Grids”, International Conference on Grid Computing,2007

[11] K.J.Arrow and G.Debreu, “Existence of an Equilibrium for a Competitive Economy”, Econometrica ,22(3), PP. 265-290 (1954).

[12] K.Ranganadhan, I.Foster, “Decoupling computation and Data Scheduling in Distributed Data Intensive Applications”, International High Performance Distributed Computing and Communication Symposium, pp 352-358,2002.

[13] K.Shahu Chatrapati, J.Ujwala Rekha, A.Vinaya Babu , “ Competitive Equilibrium Approach for Load Balancing a Computational Grid with Communication Delays”, Journal of Theoretical and Applied Information Technology, 19(2), 126-133, (2010).

[14] L.Walras, Elements d’Economie Politique Pure ; OU, Theorie de la richesse sociale. Elements of pure Economics or the Theory of Social Wealth, Lausanna, Paris(1874)

[15] Riky Subrata, Albert Y.Zomaya,”Game Theoretic Approach for Load Balancing in Computational Grids”, IEEE Transaction on Parallel and Distributed Systems, 19(1), 66-76(2008)

[16] R.M.Rahman, K.Barker, R.Barker, “A Predective Technique for Replica Selection in Grid Environment”, International Symposium

on Cluster Computing and the Grid, pp163-170,2007.

[17] S.Vazhkudai, S.Tuecke, I.Foster, "Replica Selection in the Globus Data Grid", International Symposium on Cluster Computing and the Grid, 2001.

[18] T.Braun etal, "A Comparison of Eleven Heuristics for Mapping a class of Independent Tasks on to Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing, 61(6),pp.810-837,2001.

[20] W.Hoscheck, J. Jaen-Martinez, A.Samar, H.Stockinger, and K.Stokinger, "Data Management in an International Data Grid Project", In First IEEE/ACM International Workshop on Grid on Grid Computing(Grid 2000), December 2000.

[21] Y.Cardinale, H.Casanova, "An Evaluation of Job Scheduling Strategies for Divisible Loads on Grids Platforms", HPC&S,2006.