Load Balancing in Distributed Systems: An Approach Using Cooperative Games[*]

Daniel Grosu, Anthony T. Chronopoulos
Dept. of Computer Science,
Univ. of Texas at San Antonio,
6900 N. Loop 1604 West, San Antonio, TX 78249
{dgrosu, atc}@cs.utsa.edu

Ming-Ying Leung
Dept. of Mathematics and Statistics,
Univ. of Texas at San Antonio,
6900 N. Loop 1604 West, San Antonio, TX 78249
mleung@utsa.edu

Abstract

In this paper we formulate the static load balancing problem in single class job distributed systems as a cooperative game among computers. It is shown that the Nash Bargaining Solution (NBS) provides a Pareto optimal allocation which is also fair to all jobs. We propose a cooperative load balancing game and present the structure of the NBS. For this game an algorithm for computing NBS is derived. We show that the fairness index is always 1 using NBS which means that the allocation is fair to all jobs. Finally, the performance of our cooperative load balancing scheme is compared with that of other existing schemes.

1 Introduction

The cost/performance ratio of networks of workstations has been constantly improving. This trend is expected to continue in the near future. The aggregate peak rate of such systems often matches or exceeds the peak rate offered by the fastest parallel computers. Thus, distributed computing systems are a viable and less expensive alternative to parallel computers. However, a serious difficulty in concurrent programming of a distributed system is how to deal with scheduling and load balancing of such a system which may consist of heterogeneous computers.

A distributed system can be viewed as a collection of computing and communication resources shared by active users. When the demand for computing power increases the load balancing problem becomes important. We can state the load balancing problem as follows. Given the initial job arrival rates at each computer in the system find an allocation of jobs among the computers so that the response time of the entire system over all jobs is minimized.

Often, jobs in a distributed system can be divided into different classes based on their resource usage characteristics and ownership. For example the jobs that belong to a single user can form a class. Alternatively, we can distinguish different classes of jobs by their execution times. Depending on how many job classes are considered we can have single class or multi-class job distributed systems. In this paper we consider the load balancing problem in single class job distributed systems.

There are three typical approaches to load balancing problem in single class job distributed systems:

i) *Global approach*: In this case there is only one decision maker that optimizes the response time of the entire system over all jobs and the operating point is called *social optimum*. This is the classical approach and has been studied extensively using different techniques such as nonlinear optimization [29, 30] and polymatroid optimization [25].

ii) *Cooperative approach*: In this case there are several decision makers (e.g. jobs, computers) that cooperate in making the decisions such that each of them will operate at its optimum. Decision makers have complete freedom of preplay communication to make joint agreements about their operating points. This situation can be modeled as a cooperative game and game theory offers a suitable modeling framework [6].

iii) *Noncooperative approach*: In this case each of infinitely many jobs optimizes its own response time independently of the others and they all eventually reach an equilibrium. This situation can be viewed as a noncooperative game among jobs. The equilibrium is called *Wardrop equilibrium* [8]. At the Wardrop equilibrium a job cannot receive any further benefit by changing its own decision. If the number of jobs are finite the Wardrop equilibrium reduces to the well known *Nash equilibrium* [6].

Our results
Most of the previous works on static load balancing considered as their main objective the minimization of overall expected response time. The fairness of allocation, which is

[*] This research was supported, in part, by research grants from: (1) NASA NAG 2-1383 (1999-2001); (2) State of Texas Higher Education Coordinating Board through the Texas Advanced Research/Advanced Technology Program ATP 003658-0442-1999. Some reviewer comments helped enhance the quality of presentation.

an important issue for modern distributed systems, has received relatively little attention. Our goal is to find a formal framework for characterization of fair allocation schemes that are also optimal for each job. The framework was provided by cooperative game theory. Using this framework we formulate the load balancing problem in single class job distributed systems as a cooperative game among computers. We show that the *Nash Bargaining Solution*(NBS) provides a Pareto optimal operation point for the distributed system. We give a characterization of NBS and an algorithm for computing it. We prove that the NBS is a fair solution and we compare its performance with other existing solutions. In other words the NBS guarantees the optimality and the fairness of allocation.

Related work

The problem of static load balancing in single class job distributed systems has been studied extensively. The previous studies have employed only the global and the noncooperative approach.

i) *Global approach:* The focus is on minimizing the expected response time of the entire system over all jobs. Tantawi and Towsley [30] formulated the load balancing problem as a nonlinear optimization problem and gave an algorithm for computing the allocation. Kim and Kameda [11] derived a more efficient algorithm to compute the allocation. Li and Kameda [15, 16] proposed algorithms for static load balancing in star and tree networks. Tang and Chanson [29] proposed and studied several static load balancing schemes that take into account the job dispatching strategy. Also, there exist several studies on static load balancing in multi-class job systems [10, 14, 17, 22].

ii) *Cooperative approach:* There are no known studies that involve the cooperative approach for the load balancing problem in distributed systems. This paper makes an attempt to study the cooperative approach.

iii) *Noncooperative approach:* There exist only few studies on game theoretic models and algorithms for load balancing in distributed systems. All of them involve noncooperative games. Kameda *et al*. [8] studied noncooperative games and derived load balancing algorithms for both single class and multi-class job distributed systems. For single class job systems they proposed an algorithm for computing the Wardrop equilibrium. Roughgarden [26] formulated the load balancing problem as a Stackelberg game. In this type of noncooperative game one player acts as a leader and the rest as followers. He showed that it is NP-hard to compute the optimal Stackelberg strategy and presents efficient algorithms to compute strategies inducing near-optimal solutions.

Routing traffic in networks is a closely related problem which was studied from a game theoretic perspective. Orda *et al*. [23] studied a noncooperative game in a network of parallel links with convex cost functions. They studied the existence and uniqueness of the Nash equilibrium. Altman *et al*. [2] investigated the same problem in a network of parallel links with linear cost functions. An important line of research was initiated by Koutsoupias and Papadimitriou [13], who considered a noncooperative routing game and proposed the *coordination ratio* (i.e. the ratio between the worst possible Nash equilibrium and the overall optimum) as a measure of effectiveness of the system. Mavronicolas and Spirakis [19] derived tight bounds on coordination ratio in the case of fully mixed strategies where each user assigns its traffic with non-zero probability to every link. Roughgarden and Tardos [27] showed that in a network in which the link cost functions are linear, the flow at Nash equilibrium has total latency at most 4/3 that of the overall optimal flow. They also showed that if the link cost functions are assumed to be only continuous and nondecreasing the total latency may be arbitrarily larger than the minimum possible total latency.

Recently, applications of game theory to computer science have attracted a lot of interest and have become a major trend. It is worth mentioning the recent DIMACS Workshop on Computational Issues in Game Theory and Mechanism Design [1].

Organization

The paper is structured as follows. In Section 2 we present the Nash Bargaining Solution concept for cooperative games and state several lemmas and theorems needed for our result. In Section 3 we introduce our cooperative load balancing game and derive an algorithm for computing the solution. In Section 4 the performance and fairness of our cooperative solution is compared with those of other existing solutions.

2 Cooperative Game Theory Concepts

In the following we discuss the *Nash Bargaining Solution (NBS)* [20, 21] for cooperative games. The Nash Bargaining Solution is different from the Nash Equilibrium for noncooperative games. In a cooperative game the performance of each player may be made better than the performance achieved in a noncooperative game at the Nash Equilibrium.

A cooperative game:

Assume that there are M players. Player i, $i = 1, \ldots, M$, has $f_i(x)$ as objective function. Each f_i is a function from X to $\mathbf{R}$, where $X \subseteq \mathbf{R}^L$ (L a positive integer) is a nonempty, closed and convex set, and f_i is bounded above. We want to maximize simultaneously all $f_i(x)$. Let $\mathbf{u}^0 = (u_1^0, u_2^0, \ldots, u_M^0)$ be the minimal performance required by the players without cooperation to enter the game. In other words, u_i^0 represents a minimum performance guarantee that the system must provide to the player i. $\mathbf{u}^0$ is called the initial *agreement point*.

Let $U \subset \mathbf{R}^M$ be a nonempty convex and closed set which is the set of achievable performances. Let $G = \{(U, \mathbf{u}^0) \mid U \subset \mathbf{R}^M$ is a nonempty convex and bounded set, and $\mathbf{u}^0 \in \mathbf{R}^M$ is such that $U_0 = \{\mathbf{u} \in U \mid \mathbf{u} \geq \mathbf{u}^0\}$ is nonempty $\}$. G is the set of achievable performances with respect to the initial agreement point.

Definition 2.1 Assume $x \in X$ and $\mathbf{f}(x) = (f_1(x), \dots, f_M(x))$, $\mathbf{f}(x) \in U$. Then x is said to be *Pareto optimal* if for each $x' \in X$, $f_j(x') \geq f_j(x)$, $j = 1, \dots, M$ imply $f_j(x') = f_j(x)$, $j = 1, \dots, M$. □

Remark: Pareto optimality means that it is impossible to find another point which leads to strictly superior performance for all the players.

In general, for an M-player game the set of Pareto optimal points form an $M - 1$ dimensional hypersurface consisting of an infinite number of points [24]. What is the desired operating point for our system among them? To answer this question we need additional criteria for selecting the optimal point. Such criteria are the so called fairness axioms that characterize the *Nash Bargaining Solution*.

The Nash Bargaining Solution (NBS):

Definition 2.2 [28] A mapping $S : G \to \mathbf{R}^M$ is said to be a *Nash Bargaining Solution* if:

i) $S(U, \mathbf{u}^0) \in U_0$;

ii) $S(U, \mathbf{u}^0)$ is Pareto optimal;

and satisfies the following axioms:

iii) *Linearity axiom:* If $\phi : \mathbf{R}^M \to \mathbf{R}^M$, $\phi(\mathbf{u}) = \mathbf{u}'$ with $u_j' = a_j u_j + b_j$, $a_j > 0$, $j = 1, \dots, k$ then $S(\phi(U), \phi(\mathbf{u}^0)) = \phi(S(U, \mathbf{u}^0))$.

iv) *Irrelevant alternatives axiom:* If $U \subset U'$, $(U, \mathbf{u}^0) \in G$ and $S(U', \mathbf{u}^0) \in U$, then $S(U, \mathbf{u}^0) = S(U', \mathbf{u}^0)$.

v) *Symmetry axiom:* If U is symmetrical with respect to a subset $J \subseteq \{1, \dots, M\}$ of indices (i.e. if $\mathbf{u} \in U$ and $i, j \in J$, $i < j$ imply $(u_1, \dots, u_{i-1}, u_j, u_{j+1}, \dots, u_{j-1}, u_i, u_{j+1}, \dots, u_M) \in U$) and if $u_i^0 = u_j^0$, $i, j \in J$ then $S(U, \mathbf{u}^0)_i = S(U, \mathbf{u}^0)_j$, for $i, j \in J$. □

Definition 2.3 $\mathbf{u}^*$ is a *bargaining point* if it is given by $S(U, \mathbf{u}^0)$. We call $\mathbf{f}^{-1}(\mathbf{u}^*)$ the set of *bargaining solutions*. □

Remarks: Axioms iii)-v) are called the *fairness axioms*. Essentially they say the following. The NBS is unchanged if the performance objectives are affinely scaled (Axiom iii). The bargaining point is not affected by enlarging the domain (Axiom iv). The bargaining point does not depend on the specific labels, i.e. players with the same initial points and objectives will obtain the same performance (Axiom v).

Stefanescu [28] gave the following characterization of the Nash bargaining point.

Theorem 2.1 Let X be a convex compact subset of $\mathbf{R}^L$. Let $f_i : X \to \mathbf{R}$, $i = 1, \dots, M$ be concave functions, bounded above. Let $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x}))$, $U = \{\mathbf{u} \in \mathbf{R}^M \mid \exists \mathbf{x} \in X, \mathbf{f}(\mathbf{x}) \geq \mathbf{u}\}$, $X(\mathbf{u}) = \{\mathbf{x} \in X \mid \mathbf{f}(\mathbf{x}) \geq \mathbf{u}\}$ and $X_0 = X(\mathbf{u}^0)$ be the set of strategies that enable the players to achieve at least their initial performances. Then there exists a bargaining solution and a unique bargaining point $\mathbf{u}^*$. The set of the bargaining solutions $\mathbf{f}^{-1}(\mathbf{u}^*)$ is determined as follows:

Let J be the set of players able to achieve a performance strictly superior to their initial performance, i.e. $J = \{j \in \{1, \dots, M\} \mid \exists \mathbf{x} \in X_0, f_j(\mathbf{x}) > u_j^0\}$. Each vector $\mathbf{x}$ in the bargaining solution set verifies $f_j(\mathbf{x}) > u_j^0$, for all $j \in J$ and solves the following maximization problem:

$$\max_{\mathbf{x}} \prod_{j \in J} (f_j(\mathbf{x}) - u_j^0) \qquad \mathbf{x} \in X_0 \tag{1}$$

Hence $\mathbf{u}^*$ satisfies $u_j^* > u_j^0$ for $j \in J$ and $u_j^* = u_j^0$ otherwise.

Remark: From the assumption on J, the product in equation (1) is positive. The players outside of J are not considered in the optimization problem.

Yaiche *et al.* [31] formulated an equivalent optimization problem and proved the following results. These results are needed for proving Theorem 3.1.

Proposition 2.1 If each $f_j : X \to \mathbf{R}$, $(j \in J)$ is one-to-one on $X_0 \subset X$ then $\mathbf{f}^{-1}(\mathbf{u}^*)$ is a singleton. □

Remark: Following the terminology used in [31], we call the point in this set the Nash Bargaining Solution in the rest of the paper.

Lemma 2.1 Let $g : X \to \mathbf{R}_+ \setminus \{0\}$ be a concave function, where $\mathbf{R}_+$ is the set of nonnegative real numbers. Then $h = \ln(g(.)) : X \to \mathbf{R}$ is concave. If g is one-to-one, then h is strictly concave. □

Theorem 2.2 Suppose that for each $j \in J$, $f_j : X \to \mathbf{R}$, is one-to-one on X_0. Under the assumption of Theorem 2.1, we consider the following problems:

$$(P_J): \qquad \max_{\mathbf{x}} \prod_{j \in J} (f_j(\mathbf{x}) - u_j^0) \qquad \mathbf{x} \in X_0 \tag{2}$$

$$(P_J'): \qquad \max_{\mathbf{x}} \sum_{j \in J} \ln(f_j(\mathbf{x}) - u_j^0) \quad \mathbf{x} \in X_0 \tag{3}$$

then:

1. (P_J) has a unique solution and the bargaining solution is a single point.

2. (P_J') is a convex optimization problem and has a unique solution.

3. (P_J) and (P_J') are equivalent. The unique solution of (P_J') is the bargaining solution.□

Yaiche *et al.*[31] used this equivalent problem and derived a game theoretic model for bandwidth allocation and pricing in broadband networks.

3 Load balancing as a cooperative game among computers

We consider a single class job distributed system that consists of n heterogeneous computers. We consider a game in which each computer is a player and it must minimize the expected execution time of jobs that it processes. If $F_i(\beta_i)$ denotes the expected execution time of jobs processed at computer i, we can express the game as follows:

$$\min_{\beta_i} F_i(\beta_i), \qquad i = 1, \ldots, n \tag{4}$$

where β_i is the average arrival rate of jobs at computer i.

Modeling each computer as an M/M/1 queueing system [18], $F_i(\beta_i) = \frac{1}{\mu_i - \beta_i}$, where μ_i is the average processing rate of computer i. The minimization problem, associated with the above game, becomes:

$$\min_{\beta_i} \frac{1}{\mu_i - \beta_i}, \qquad i = 1, \ldots, n \tag{5}$$

subject to:

$$\beta_i < \mu_i, \qquad i = 1, \ldots, n \tag{6}$$

$$\sum_{i=1}^{n} \beta_i = \Phi \tag{7}$$

$$\beta_i \geq 0, \qquad i = 1, \ldots, n \tag{8}$$

where Φ is the total job arrival rate of the system. The first constraint is the ‘stability’ condition and the second one is the ‘conservation law’ for the M/M/1 system.

This problem is equivalent to:

$$\max_{\beta_i} (-\beta_i), \qquad i = 1, \ldots, n \tag{9}$$

subject to:

$$-\beta_i > -\mu_i, \qquad i = 1, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n} \beta_i = \Phi \tag{11}$$

$$-\beta_i \leq 0, \qquad i = 1, \ldots, n \tag{12}$$

Based on this optimization problem we can formulate an equivalent game as follows. The n computers are the players, each having $f_i(\beta_i) = -\beta_i$ as objective functions. All players need to maximize their objective functions simultaneously. The set X is defined by the constraints:

$$-\beta_i \geq -\mu_i, \qquad i = 1, \ldots, n \tag{13}$$

$$\sum_{i=1}^{n} \beta_i = \Phi \tag{14}$$

$$-\beta_i \leq 0, \qquad i = 1, \ldots, n \tag{15}$$

Remark: To satisfy the compactness requirement for X we allow the possibility that $\beta_i = \mu_i$. This requirement will be dropped in the following.

We assume that all n computers in the set J of players are able to achieve performance strictly superior to the initial performance. The initial performance is given by $\beta_i = \mu_i$, which corresponds to $f_i = -\mu_i$, ($i = 1, \ldots, n$). We assume that the initial agreement point is $u_i^0 = -\mu_i$, $i = 1, \ldots, n$. In other words all computers agree that they will choose β_i such that $-\beta_i > -\mu_i$. This is also the ’stability’ condition for the M/M/1 system.

Our results are given in Theorem 3.1 - 3.3 and Proposition 3.1. Their proofs are presented in the Appendix.

Theorem 3.1 For the load balancing cooperative game defined above the bargaining solution is determined by solving the following optimization problem:

$$\max_{\beta} \sum_{i=1}^{n} \ln(\mu_i - \beta_i) \tag{16}$$

subject to:

$$\beta_i < \mu_i, \qquad i = 1, \ldots, n \tag{17}$$

$$\sum_{i=1}^{n} \beta_i = \Phi \tag{18}$$

$$\beta_i \geq 0, \qquad i = 1, \ldots, n \tag{19}$$

□

As a first step in obtaining the solution for our load balancing cooperative game we solve the optimization problem given in Theorem 3.1 without requiring β_i ($i = 1, \ldots, n$) be non-negative. The solution of this problem is given in the following proposition.

Proposition 3.1 The solution of the optimization problem in Theorem 3.1 without the constraint $\beta_i \geq 0, i = 1, \ldots, n$ is given by:

$$\beta_i = \mu_i - \frac{\sum_{j=1}^{n} \mu_j - \Phi}{n} \tag{20}$$

□

In practice we cannot use this solution because there is no guarantee that β_i ($i = 1, \ldots, n$) is always non-negative. Note that β_k is negative when $\mu_k < \frac{\sum_{j=1}^{n} \mu_j - \Phi}{n}$. This means that computer k is very slow. In such cases we make the solution feasible by setting $\beta_k = 0$ and remove computer k from the system. Setting β_k equal to zero means that we do not assign jobs to the extremely slow computers.

Assuming that computers are ordered in decreasing order of their processing rates, we eliminate the slowest computer and recompute the allocation for a system with $n-1$ computers. This procedure is applied until a feasible solution is found.

Based on this fact and Proposition 3.1, we derive an algorithm (called COOP) for obtaining the Nash Bargaining Solution for the load balancing cooperative game. In the following we present this algorithm:

COOP algorithm:

```
Input: Average processing rates: $\mu_1, \mu_2, \dots \mu_n$;
          Total arrival rate: $\Phi$
Output: Load allocation: $\beta_1, \beta_2, \dots \beta_n$;
          1. Sort the computers in decreasing order of
             their average processing rates
             ($\mu_1 \geq \mu_2 \geq \dots \geq \mu_n$);
          2. $c \leftarrow \frac{\sum_{j=1}^{n} \mu_j - \Phi}{n}$;
          3. while ( $c > \mu_n$ ) do
                 $\beta_n \leftarrow 0$;
                 $n \leftarrow n - 1$;
                 $c \leftarrow (c - \frac{\mu_{n+1}}{n+1}) \frac{n+1}{n}$;
          4. for $i = 1, \dots, n$ do
                 $\beta_i \leftarrow \mu_i - c$;
```

The following theorem proves the correctness of this algorithm.

Theorem 3.2 The allocation $\{\beta_1, \beta_2, \dots, \beta_n\}$ computed by the COOP algorithm solves the optimization problem in Theorem 3.1 and is the NBS for our cooperative game. □

The execution time of this algorithm is in $O(n \log n)$. In general determining the NBS is an NP-hard problem [5]. In our case we were able to obtain an $O(n \log n)$ algorithm because the cooperative load balancing game is a convex game (i.e. player's objective functions are convex).

The *fairness index*

$$I(\mathbf{T}) = \frac{[\sum_{i=1}^{n} T_i]^2}{n \sum_{i=1}^{n} T_i^2} \tag{21}$$

was proposed in [7] to quantify the fairness of load balancing schemes. Here the parameter $\mathbf{T}$ is the vector $\mathbf{T} = (T_1, T_2, \dots, T_n)$ where T_i is the average execution time for jobs that are processed at computer i.
Remark: This index is a measure of the 'equality' of execution times at different computers. So it is a measure of load balance. If all the computers have the same expected job execution times then $I = 1$ and the system is 100% fair to all jobs and is load balanced. If the differences on T_i increase, I decreases and the load balancing scheme favors only some tasks.

For the proposed load balancing cooperative game we can state the following:

Theorem 3.3 The fairness index equals 1 when we use the Nash Bargaining Solution for the proposed load balancing cooperative game. □

This means that all jobs receive a fair treatment independent of the allocated computer.

4 Experimental Results

4.1 Simulation Environment

The simulations were carried out using Sim++ [4], a simulation software package written in C++. This package provides an application programming interface which allow the programmer to call several functions related to event scheduling, queueing, preemption and random number generation. The simulation model consists of a collection of computers connected by a communication network. Jobs arriving at the system are distributed by a central dispatcher to the computers according to the specified load balancing scheme. Jobs which have been dispatched to a particular computer are *run-to-completion* (i.e. no preemption) in FCFS (first-come-first-served) order.

Each computer is modeled as an M/M/1 queueing system [12]. The main performance metrics used in our simulations are the *expected response time* and the *fairness index*. The simulations were run over several millions of seconds, sufficient to generate a total of 1 to 2 millions jobs typically. Each run was replicated five times with different random number streams and the results averaged over replications. The standard error is less than 5% at the 95% confidence level.

4.2 Performance Evaluation

For comparison purposes we consider both static and dynamic load balancing schemes. In addition to our cooperative static scheme (COOP) we implemented one dynamic and three static schemes. A brief description of these schemes is given below:

- **Static Schemes**:

  - Proportional Scheme (PROP) [3]: According to this scheme jobs are allocated in proportion to the processing speed of computers. The following algorithm is used for obtaining the load allocation.

  PROP algorithm:

```
Input: Average processing rates: $\mu_1, \mu_2, \dots \mu_n$;
          Total arrival rate: $\Phi$;
Output: Load allocation: $\beta_1, \beta_2, \dots \beta_n$;
          for $i = 1, \dots, n$ do
              $\beta_i \leftarrow \Phi \frac{\mu_i}{\sum_{j=1}^{n} \mu_j}$;
```

This allocation seems to be a natural choice but it may not minimize the average response time of the system and is unfair.

- Overall Optimal Scheme (OPTIM) [29, 30]: This scheme minimizes the expected execution time over all jobs executed by the system. The loads at each computer (β_i) are obtained by solving the following nonlinear optimization problem:

$$\min \frac{1}{\Phi} \sum_{i=1}^{n} \beta_i F_i(\beta_i) \qquad (22)$$

subject to the constraints (6-8). The following algorithm is used for obtaining the load allocation.

OPTIM algorithm:

```
Input: Average processing rates: $\mu_1, \mu_2, \ldots \mu_n$;
    Total arrival rate: $\Phi$;
Output: Load allocation: $\beta_1, \beta_2, \ldots \beta_n$;
    1. Sort the computers in decreasing order of
       their average processing rates
       ($\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$);
    2. $c \leftarrow \frac{\sum_{i=1}^{n} \mu_i - \Phi}{\sum_{i=1}^{n} \sqrt{\mu_i}}$;
    3. while ( $c > \sqrt{\mu_n}$ ) do
           $\beta_n \leftarrow 0$;
           $n \leftarrow n - 1$;
           $c \leftarrow \frac{\sum_{i=1}^{n} \mu_i - \Phi}{\sum_{i=1}^{n} \sqrt{\mu_i}}$;
    4. for $i = 1, \ldots, n$ do
           $\beta_i \leftarrow \mu_i - c\sqrt{\mu_i}$;
```

This scheme provides the overall optimum for the expected execution time but is unfair.

- Wardrop Equilibrium Scheme (WARDROP) [8]: In this scheme each of infinitely many jobs optimizes its response time for itself independently of others. In general the Wardrop equilibrium solution is not Pareto optimal and in some cases we expect worse response time than the other policies [8]. It is based on an iterative procedure that is not very efficient. For a complete description of WARDROP algorithm see [8]. The advantage of this scheme is that it provides a fair allocation.

- **Dynamic Scheme**:

- Shortest Expected Delay Scheme (DYNAMIC) [3]: This is a centralized scheme in which a newly arrived job is assigned to the computers that yield the minimum expected delay. The expected delay of computer i is estimated using the equation: $D_i = \frac{q_i+1}{\mu_i}$, where q_i is the queue length of computer i and μ_i is the average processing rate of computer i. The following algorithm is executed by the dispatcher when a new job arrives in the system.

DYNAMIC algorithm:

```
1. Find $\min\{D_i | i = 1, \ldots, n\}$.
2. If minimum is not unique
       then select $k$ such that $\mu_k$ is the maximum
       among ties.
3. Assign the new job to computer $k$.
```

The expected execution time obtained by this scheme is used in our comparisons as a lower bound to the execution times of the static schemes.

Remark: Among the four schemes described above, the WARDROP scheme is the only scheme that is based on game theoretic concepts.

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

4.2.1 Effect of System Utilization

To study the effect of system utilization we simulated a heterogeneous system consisting of 16 computers with four different processing rates. In Table 1 we present the system configuration. The first row contains the relative processing rates of each of the four computer types. Here, the relative processing rate for computer C$_i$ is defined as the ratio of the processing rate of C$_i$ to the processing rate of the slowest computer in the system. The second row contains the number of computers in the system corresponding to each computer type. The last row shows the processing rate of each computer type in the system. We choose 0.013 jobs/sec. as the processing rate for the slowest computer because it is a value that can be found in real distributed systems [29]. Also we consider only computers that are at most ten times faster than the slowest because this is the case in most of the current heterogeneous distributed systems.

Relative processing rate	1	2	5	10
Number of computers	6	5	3	2
Processing rate (jobs/sec)	0.013	0.026	0.065	0.13

Table 1. System configuration.

In Figure 1 we present the expected response time of the system for different values of system utilization (ranging from 10% to 90%). *System utilization* (ρ) is defined as the ratio of total arrival rate to aggregate processing rate of the system: $\rho = \frac{\Phi}{\sum_{i=1}^{n} \mu_i}$. It can be observed that at low loads (ρ from 10% to 40%) all the schemes except PROP yield almost the same performance. The poor performance of PROP scheme is due to the fact that the less powerful computers are significantly overloaded.

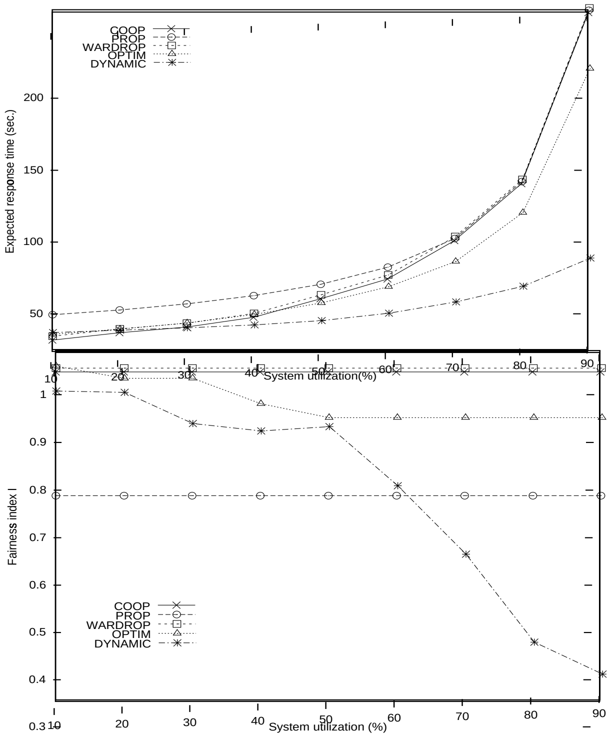

Figure 1. The expected response time and fairness index vs. system utilization.

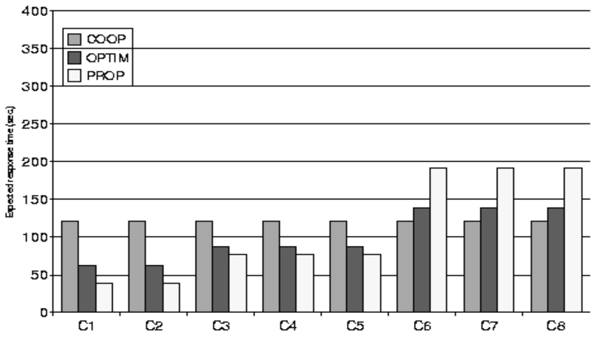

Figure 2. The expected response time at computer 1 to 8 (high system load).

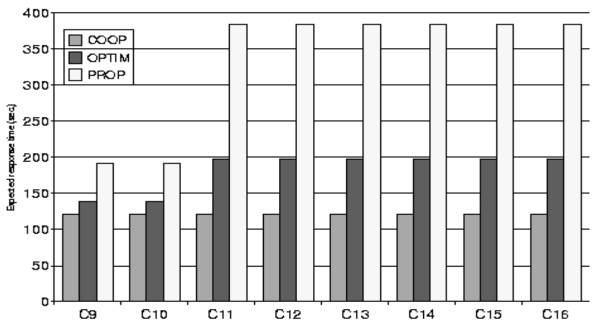

Figure 3. The expected response time at computer 9 to 16 (high system load).

At medium loads (ρ from 40% to 60%) the COOP scheme performs significantly better than PROP and approaches the performance of OPTIM. For example at load level of 50% the mean response time of COOP is 19% less than PROP and 20% greater than OPTIM.

At high loads COOP and PROP yield the same expected response time which is greater than that of OPTIM. The DYNAMIC scheme outperforms all static schemes under heavy loads. This implies that at high load the dynamic scheme is effective.

Remark: The WARDROP and COOP yield the same performance for the whole range of system utilization and they are also 100% fair. The reason for this is that the noncooperative game used to derive the Wardrop equilibrium is a convex game and it has a unique equilibrium that is a Pareto optimal solution [9]. The advantage of COOP scheme is that the algorithm for computing the allocation is very simple. The WARDROP scheme involves a more complicated algorithm that require an iterative process that take more time to converge. We ran both algorithms for a system of 16 computers on a SUN 10 (440 MHz) workstation and we obtained the following execution times: 4.1 msec for WARDROP and 0.8 msec for COOP.

An important performance metric is the fairness index. The COOP and WARDROP schemes maintain a fairness index of 1 over the whole range of system loads and they are the only fair schemes here. It can be shown that the PROP has a fairness index of 0.731 which is a constant independent of the system load. The fairness index of OPTIM varies from 1 at low load, to 0.88 at high load.

An interesting issue is the impact of static load balancing schemes on individual computers. In Figure 2 and 3 we present the expected response time at each computer in the system when the system is heavily loaded ($\rho = 80\%$). The WARDROP scheme gives the same results as COOP and is not presented in these figures. The COOP scheme guarantees the same execution time at each computer and utilizes all the computers. The value of the expected response time for each computer is equal to the value of overall expected response time. The difference in the expected response time between the fastest and slowest computers is huge in the case of PROP (350 sec.) and moderate in the case of OPTIM (130 sec.).

In the case of PROP and OPTIM jobs are treated unfairly in the sense that a job allocated to a fast computer will have a low expected response time and a job allocated to a slow computer will have a high expected response time. COOP scheme provides really a fair and load balanced allocation and in some systems this is a desirable property.

4.2.2 Effect of heterogeneity

In a distributed system, heterogeneity usually consists of: processor speed, memory and I/O. A simple way to characterize system heterogeneity is to use the processor speed. Furthermore it is reasonable to assume that a computer with high speed processor will have matching resources (memory and I/O). One of the common measures of heterogeneity is the *speed skewness* [29] which is defined as the ratio of maximum processing rate to minimum processing rate of the computers. This measure is somehow limited but for our goals it is satisfactory.

In this section we investigate the effectiveness of load balancing schemes by varying the speed skewness. We simulate a system of 16 heterogeneous computers: 2 fast and 14 slow. The slow computers have a relative processing rate of 1 and we varied the relative processing rate of the fast computers from 1 (which correspond to a homogeneous system) to 20 (which correspond to a highly heterogeneous system). The system utilization was kept constant $\rho = 60\%$.

In Figure 4 we present the effect of speed skewness on the expected response time and fairness. It can be observed that increasing the speed skewness the OPTIM and COOP schemes yield an expected response time close to that of DYNAMIC which means that in highly heterogeneous systems the COOP and OPTIM are very effective. COOP has the additional advantage of a fair allocation which is very important in some systems. PROP scheme performs poorly because it overloads the slowest computers.

5 Conclusion

In this paper we have presented a game theoretic framework for obtaining a fair load balancing scheme. The main goal was to derive a fair and optimal allocation scheme. We formulated the load balancing problem in single class job distributed systems as a cooperative game among computers. We showed that the Nash Bargaining Solution (NBS) of this game provides a Pareto optimal operation point for the distributed system and it is also a fair solution. For the proposed cooperative load balancing game we presented the structure of the NBS. Based on this structure we derived a new algorithm for computing it. We showed that the fairness index is always 1 using our new cooperative load balancing scheme, which means that the solution is fair to all jobs. We compared the performance of our cooperative load balancing scheme with other existing schemes.

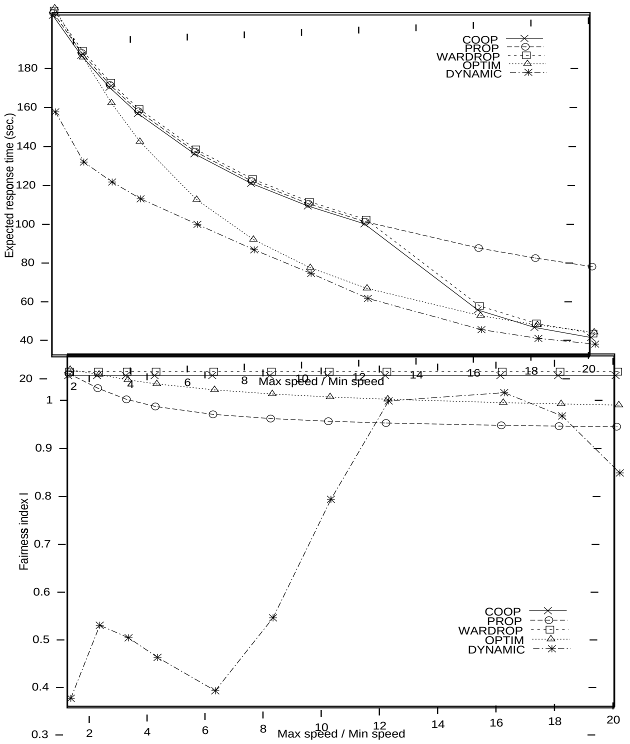

Figure 4. The effect of heterogeneity on the expected response time and fairness index.

Future work will address the development of game theoretic models for load balancing in the context of uncertainty as well as game theoretic models for dynamic load balancing.

References

[1] *DIMACS Workshop on Computational Issues in Game Theory and Mechanism Design,* November 2001, DIMACS Center, Rutgers University. `http://dimacs.rutgers.edu/Workshops/gametheory/`.

[2] E. Altman, T. Basar, T. Jimenez, and N. Shimkin. Routing in two parallel links: Game-theoretic distributed algorithms. *to appear in: Journal of Parallel and Distributed Computing*, 2001.

[3] Y. C. Chow and W. H. Kohler. Models for dynamic load balancing in a heterogeneous multiple processor system. *IEEE Trans. Comput.*, C-28(5):354–361, May 1979.

[4] R. M. Cubert and P. Fishwick. *Sim++ Reference Manual.* CISE, Univ. of Florida, July 1995.

[5] X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, May 1994.

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

[7] R. K. Jain, D.M. Chiu, and W. R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Technical Report DEC-TR-301, Digital Equipment Corporation, Eastern Research Lab, 1984.

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

[9] H. Kameda and Y. Zhang. Uniqueness of the solution for optimal static routing in open BCMP queueing networks. *Mathl. Comput. Modelling*, 22(10-12):119–130, Nov.-Dec. 1995.

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

[11] C. Kim and H. Kameda. An algorithm for optimal static load balancing in distributed computer systems. *IEEE Trans. Comput.*, 41(3):381–384, March 1992.

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

[13] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proc. of the 16th Annual Symp. on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[14] H. Lee. Optimal static distribution of prioritized customers to heterogeneous parallel servers. *Computers Ops. Res.*, 22(10):995–1003, December 1995.

[15] J. Li and H. Kameda. A decomposition algorithm for optimal static load balancing in tree hierarchy network configuration. *IEEE Trans. Parallel and Distributed Syst.*, 5(5):540–548, May 1994.

[16] J. Li and H. Kameda. Optimal static load balancing in star network configurations with two-way traffic. *J. of Parallel and Distributed Computing*, 23(3):364–375, December 1994.

[17] J. Li and H. Kameda. Load balancing problems for multiclass jobs in distributed/parallel computer systems. *IEEE Trans. Comput.*, 47(3):322–332, March 1998.

[18] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, Mass., 1984.

[19] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proc. of the 33rd Annual ACM Symp. on Theory of Computing*, pages 510–519, July 2001.

[20] A. Muthoo. *Bargaining Theory with Applications*. Cambridge Univ. Press, Cambridge, U.K., 1999.

[21] J. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, April 1950.

[22] L. M. Ni and K. Hwang. Adaptive load balancing in a multiple processor system with many job classes. *IEEE Trans. Software Eng.*, SE-11(5):491–496, May 1985.

[23] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Trans. Networking*, 1(5):510–521, October 1993.

[24] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proc. of the 41th IEEE Symp. on Foundations of Computer Science*, pages 86–92, November 2000.

[25] K. W. Ross and D. D. Yao. Optimal load balancing and scheduling in a distributed computer system. *Journal of the ACM*, 38(3):676–690, July 1991.

[26] T. Roughgarden. Stackelberg scheduling strategies. In *Proc. of the 33rd Annual ACM Symp. on Theory of Computing*, pages 104–113, July 2001.

[27] T. Roughgarden and E. Tardos. How bad is selfish routing? In *Proc. of the 41th IEEE Symp. on Foundations of Computer Science*, pages 93–102, November 2000.

[28] A. Stefanescu and M. V. Stefanescu. The arbitrated solution for multi-objective convex programming. *Rev. Roum. Math. Pure Appl.*, 29:593–598, 1984.

[29] 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.

[30] A. N. Tantawi and D. Towsley. Optimal static load balancing in distributed computer systems. *Journal of the ACM*, 32(2):445–465, April 1985.

[31] H. Yaiche, R. R. Mazumdar, and C. Rosenberg. A game theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Trans. Networking*, 8(5):667–678, October 2000.

A Appendix

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

Proof of Theorem 3.1:
In Theorem 2.1 we consider $f_i(\beta_i) = -\beta_i$ which are concave and bounded above. The set defined by the constraints is convex and compact. The conditions of Theorem 2.1 are satisfied. Using the result of Theorem 2.1 and the fact that $f_i(\beta_i) = -\beta_i$ are one-to-one functions of β_i and applying Theorem 2.2 the results follows. □

Proof of Proposition 3.1:
The constraints in Theorem 3.1 are linear in β_i and $f_i(\beta_i) = -\beta_i$ has continuous first partial derivatives. This implies that the first order Kuhn-Tucker conditions are necessary and sufficient for optimality [18].

Let $\alpha \leq 0$, $\eta_i \leq 0$, $i = 1, \ldots, n$ denote the Lagrange multipliers [18]. The Lagrangian is:

$$L(\beta_i, \alpha, \eta_i) = \sum_{i=1}^{n} \ln(\mu_i - \beta_i) + \alpha(\sum_{i=1}^{n} \beta_i - \Phi)+$$

$$\sum_{i=1}^{n} \eta_i(\beta_i - \mu_i) \tag{23}$$

The first order Kuhn-Tucker conditions are:

$$\frac{\partial L}{\partial \beta_i} = -\frac{1}{\mu_i - \beta_i} + \alpha + \eta_i = 0, \qquad i = 1, \ldots, n \tag{24}$$

$$\frac{\partial L}{\partial \alpha} = \sum_{i=1}^{n} \beta_i - \Phi = 0 \tag{25}$$

$$\mu_i - \beta_i \geq 0,\ \eta_i(\mu_i - \beta_i) = 0,\ \eta_i \leq 0,\ i = 1, \ldots, n \tag{26}$$

We have $\mu_i - \beta_i > 0, \quad i = 1, \ldots, n$. This implies that $\eta_i = 0, \quad i = 1, \ldots, n$.

The solution of these conditions is:

$$\frac{1}{\mu_i - \beta_i} - \alpha = 0\ , \qquad i = 1, \ldots, n \tag{27}$$

$$\sum_{i=1}^{n} \beta_i = \Phi \tag{28}$$

It then follows that:

$$\beta_i = \mu_i - \frac{\sum_{j=1}^{n} \mu_j - \Phi}{n} \tag{29}$$

□

Lemma A.1 Let $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_m$. If $\mu_m < \frac{\sum_{j=1}^{m} \mu_j - \Phi}{m}$ then the objective function from Theorem 3.1 is maximized, subject to the extra constraint $\beta_m \geq 0$, when $\beta_m = 0$. □

Proof: We add a new condition to the first order conditions (24-26). The new set of conditions is as follows:

$$\frac{\partial L}{\partial \beta_i} = -\frac{1}{\mu_i - \beta_i} + \alpha + \eta_i = 0\ , \qquad i = 1, \ldots m - 1 \tag{30}$$

$$\frac{\partial L}{\partial \beta_m} = -\frac{1}{\mu_m - \beta_m} + \alpha + \eta_m + \gamma_m = 0 \tag{31}$$

$$\frac{\partial L}{\partial \alpha} = \sum_{i=1}^{m} \beta_i - \Phi = 0 \tag{32}$$

$$\mu_i - \beta_i \geq 0,\ \eta_i(\mu_i - \beta_i) = 0,\ \eta_i \leq 0,\ i = 1, \ldots m \tag{33}$$

$$\beta_m \geq 0, \quad \gamma_m \beta_m = 0, \quad \gamma_m \leq 0 \tag{34}$$

We have $\mu_i - \beta_i > 0, \quad i = 1, \ldots m$. This implies that $\eta_i = 0, \quad i = 1, \ldots m$.

$$\frac{1}{\mu_i - \beta_i} - \alpha = 0\ , \qquad i = 1, \ldots m - 1 \tag{35}$$

$$\frac{1}{\mu_m - \beta_m} - \alpha + \gamma_m = 0 \tag{36}$$

$$\sum_{i=1}^{m} \beta_i = \Phi \tag{37}$$

$$\beta_m \geq 0, \quad \gamma_m \beta_m = 0, \quad \gamma_m \leq 0 \tag{38}$$

From equation (38) the value of β_i is either zero or positive. We consider each case separately.

Case 1: $\beta_m > 0$. Equation (38) implies $\gamma_m = 0$ and the solution of the conditions is:

$$\beta_i = \mu_i - \frac{\sum_{j=1}^{m} \mu_j - \Phi}{m} \quad i = 1, \ldots m \tag{39}$$

Case 2: $\beta_m = 0$. It follows from equation (38) that $\gamma_m \leq 0$.

The restriction $\beta_m \geq 0$ becomes active when $\frac{1}{\mu_m - \beta_m} - \alpha = -\gamma_m > 0$ which implies that $\mu_m < \frac{\sum_{j=1}^{m} \mu_j - \Phi}{m}$ and lemma is proved. □

Proof of Theorem 3.2:
The while loop in step 3 finds the minimum index m for which $\mu_m < \frac{\sum_{j=1}^{m} \mu_j - \Phi}{m}$.

If $m = n$ (that means that all β_i are positive) we apply Proposition 3.1 and the result follows.

If $m < n$ we have $\mu_i < \frac{\sum_{j=1}^{i} \mu_j - \Phi}{i}$ for $i = m, \ldots, n$. According to Lemma A.1 $\beta_m, \ldots, \beta_n$ must be 0 in order to maximize the objective function from Theorem 3.1. The algorithm sets $\beta_i = 0$ for $i = m, \ldots, n$ in the while loop.

Because $\mu_m < \frac{\sum_{j=1}^{m} \mu_j - \Phi}{m}$ implies $0 < \mu_m m \leq \sum_{j=1}^{m} \mu_j - \Phi$ and then $\Phi < \sum_{j=1}^{m} \mu_j$, we can apply Proposition 3.1 to the first m indices (that corresponds to the first m fast computers) and the values of the load allocation $\{\beta_1, \beta_2, \ldots, \beta_m\}$ maximizes the objective function and are given by:

$$\beta_i = \mu_i - \frac{\sum_{j=1}^{m} \mu_j - \Phi}{m}, \quad i = 1, \ldots m \tag{40}$$

This is done in step 4. All these β_i, for $i = 1, \ldots m$ are guaranteed to be nonnegative because $\mu_i > \frac{\sum_{j=1}^{m} \mu_j - \Phi}{m}$. The load allocation $\{\beta_1, \beta_2, \ldots, \beta_m\}$ is the solution of the optimization problem in Theorem 3.1. According to Theorem 3.1 this is also the NBS of our cooperative load balancing game. □

Proof of Theorem 3.3:
Using Proposition 3.1, T_i for all n_a allocated computers ($\beta_i \neq 0, i = 1, \ldots, n_a$) can be expressed as:

$$T_i = \frac{1}{(\mu_i - \beta_i)} = \frac{n_a}{\sum_{j=1}^{n_a} \mu_j - \Phi} \tag{41}$$

Thus all $T_i, i = 1 \ldots, n_a$ are equal and this implies $I(\mathbf{T}) = 1$. □