Cooperative Load Balancing for a Network of Heterogeneous Computers

Satish Penmatsa and Anthony T. Chronopoulos

The University of Texas at San Antonio
Dept. of Computer Science
6900 N Loop, 1604 W, San Antonio, Texas 78249, USA
{spenmats, atc}@cs.utsa.edu

Abstract

In this paper we present a game theoretic approach to solve the static load balancing problem in a distributed system which consists of heterogeneous computers connected by a single channel communication network. We use a cooperative game to model the load balancing problem. Our solution is based on the Nash Bargaining Solution (NBS) which provides a Pareto optimal solution for the distributed system and is also a fair solution. An algorithm for computing the NBS is derived for the proposed cooperative load balancing game. Our scheme is compared with that of other existing schemes under simulations with various system loads and configurations. We show that the solution of our scheme is near optimal and is superior to the other schemes in terms of fairness.

1. Introduction

In this paper, we consider the static load balancing problem for single class jobs in a distributed computer system that consists of heterogeneous host computers (nodes) interconnected by a single channel communication network. Load balancing is achieved by transferring some jobs from nodes that are heavily loaded to those that are idle or lightly loaded. A communication delay will be incurred as a result of sending a job to a different computer for processing.

The load balancing problem is formulated as a cooperative game among the computers and the communication subsystem and game theory offers a suitable modeling framework [2]. 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) which provides a Pareto optimal and fair solution, we provide an algorithm for computing the NBS for our cooperative load balancing game.

Past work on load balancing jobs considered optimization of the entire system expected response time [17, 7, 9, 16, 6] or applied game theory without taking into account the communication subsystem [4, 3, 14].

The main goal of our load balancing scheme is to provide fairness to all the jobs, i.e. all the jobs should experience the same expected response time independent of the allocated computer. The fairness of allocation is an important factor in modern distributed systems and our scheme will be suitable for systems in which the fair treatment of the users' jobs is as important as other performance characteristics. We show that our cooperative load balancing scheme not only provides fairness but also provides a Pareto optimal operating point for the entire system. We make simulations with various system loads and configurations to evaluate the performance of our cooperative load balancing scheme.

2. Cooperative Game Theory Concepts

In this section, we summarize some concepts and results from cooperative game theory which are used in the sequel.

***Definition* 2.1 (A cooperative game)** *A cooperative game consists of:*

- N players;
- *A nonempty, closed and convex set* $X \subseteq \mathbf{R}^N$ *which is the* set of strategies *of the* N *players.*

1-4244-0054-6/06/$20.00 ©2006 IEEE

- *For each player i, $i = 1, 2, \ldots, N$, an* objective function f_i. *Each f_i is a function from X to $\mathbf{R}$ and it is bounded below. The goal is to minimize simultaneously all $f_i(X)$.*
- *For each player i, $i = 1, 2, \ldots, N$, a* minimal value *of f_i, denoted u_i^0, required by player i without any cooperation to enter the game. The vector $\mathbf{u}^0 = (u_1^0, u_2^0, \ldots, u_N^0)$ is called the initial* agreement point.

We are interested in finding solutions for the cooperative game defined above that are Pareto optimal.

***Definition* 2.2 (Pareto optimality)** *[11] The point $u \in U$ is said to be Pareto optimal if for each $v \in U$, $v \leq u$, then $v = u$. Here U, $U \subset \mathbf{R}^N$ is the set of achievable performances [18].*

***Definition* 2.3 (The Nash Bargaining Solution (NBS))** *[12, 13, 15] A mapping $S : G \rightarrow \mathbf{R}^N$ is said to be a NBS if:*

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

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

and satisfies the fairness axioms [13]. Here G denotes the set of achievable performances with respect to the initial agreement point [18].

***Definition* 2.4 (Bargaining point)** *[15]* $\mathbf{u}^*$ *is a* (Nash) bargaining point *if it is given by $S(U, \mathbf{u}^0)$ and $\mathbf{f}^{-1}(\mathbf{u}^*)$ is called the set of* (Nash) bargaining solutions.

The following characterization of the Nash bargaining point forms the basis for the results in the sequel.

***Theorem* 2.1 (Nash bargaining point characterization)** *[15, 18] Consider the assumptions from the above definitions and references therein. Let J denote the set of players who are able to achieve a performance strictly superior to their initial performance and let X_0 denote the set of strategies that enable the players to achieve at least their initial performances. Let the vector function $\{f_j\}, j \in J$ be one-to-one on X_0. Then, there exists a unique bargaining point $\mathbf{u}^*$ and the set of the bargaining solutions $\mathbf{f}^{-1}(\mathbf{u}^*)$ is determined by solving the following optimization problems:*

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

$$(P_J'): \qquad \min_{\mathbf{x}} \sum_{j \in J} \ln(f_j(\mathbf{x}) - u_j^0) \quad \mathbf{x} \in X_0 \qquad (2)$$

Then, (P_J) and (P_J') are equivalent. (P_J') is a convex optimization problem and has a unique solution. The unique solution of (P_J') is the bargaining solution. □

3. System Model

We consider a distributed system model with n nodes (or computers) connected by a single channel communication network as shown in Figure 1. The computers and the communication network are assumed to be product-form queuing network models.

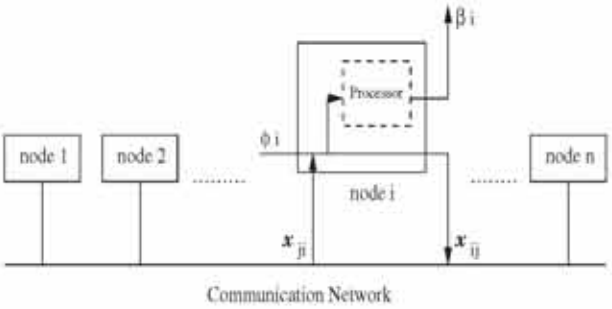

Figure 1. Distributed System Model

Terminology and Assumptions:

(i) ϕ_i denotes the external job arrival rate at node i. The total external job arrival rate of the system is denoted by $\Phi = \sum_{i=1}^{n} \phi_i$. (ii) The job processing rate (load) at node i is denoted by β_i. (iii) x_{ij} denotes the job flow rate from node i to node j. (iv) A job arriving at node i may be either processed at node i or transferred to another node j through the communication network. The decision of transferring a job does not depend on the state of the system and hence is *static* in nature. (v) A job transferred from node i to node j receives its service at node j and is not transferred to other nodes. If a node i sends (receives) jobs to (from) node j, node j does not send (receive) jobs to (from) node i.

(vi) The response time of a job in a system as above consists of a node delay (queuing and processing delays) at the processing node and also some possible communication delay incurred due to a job transfer. We denote the mean node delay or the expected response time of a job at node i by $D_i(\beta_i)$. Modeling each node as an M/M/1 queuing system gives [8, 17]:

$$D_i(\beta_i) = \frac{1}{\mu_i - \beta_i}, \quad i = 1, \ldots, n. \qquad (3)$$

where μ_i is the processing (service) rate of computer i.

(vii) We assume that the expected communication delay from node i to node j is independent of the source-destination pair (i, j) but may depend on the total traffic through the network denoted by λ where $\lambda = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}$. We denote the mean communication delay for a job by $G(\lambda)$. Modeling the communication network as an M/M/1 queuing system gives

[8, 17]:

$$G(\lambda) = \frac{t}{1 - t\lambda}, \quad \lambda < \frac{1}{t} \tag{4}$$

where t is the mean communication time for sending and receiving a job. Note that $D_i(\beta_i)$ and $G(\lambda)$ are increasing positive functions.

(viii) we classify the nodes in the following way as in [17]:

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

The network traffic λ can be expressed in terms of the variable β_i as

$$\lambda = \frac{1}{2}\sum_{i=1}^{n} |\phi_i - \beta_i| \tag{5}$$

(ix) We define the following functions:

$$d_i(\beta_i) = \frac{\partial}{\partial \beta_i} \ln D_i(\beta_i) = \frac{1}{\mu_i - \beta_i} \tag{6}$$

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

$$d_i^{-1}(x) = \begin{cases} \mu_i - \frac{1}{x}, & \text{if } x > \frac{1}{\mu_i} \\ 0, & \text{if } x \leq \frac{1}{\mu_i} \end{cases} \tag{8}$$

4. Cooperative Load Balancing

In this section, we formulate the load balancing problem as a cooperative game among the computers and the communication network. We consider an $n+1$ player game where the n computers try to minimize their expected response time $D_i(\beta_i)$ and the $(n+1)th$ player, the communication subsystem, tries to minimize the expected communication delay $G(\lambda)$. So, the objective function for each computer i, $i = 1, \ldots, n$ can be expressed as:

$$f_i(X) = D_i(\beta_i) \tag{9}$$

and the objective function for the communication subsystem can be expressed as:

$$f_{n+1}(X) = G(\lambda) \tag{10}$$

where $X = [\beta_1, \ldots, \beta_n, \lambda]^T$ is the set of strategies of the $n+1$ players.

***Definition* 4.1 (The cooperative load balancing game)** *The cooperative load balancing game consists of:*

- *n computers and the communication subsystem as* players*;*
- *The* set of strategies, *X, is defined by the following constraints:*

$$\beta_i < \mu_i, \qquad i = 1, \ldots, n \tag{11}$$

$$\sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n} \phi_i = \Phi, \tag{12}$$

$$\beta_i \geq 0, \qquad i = 1, \ldots, n \tag{13}$$

- *For each computer i, $i = 1, \ldots, n$, the* objective function *$f_i(X) = D_i(\beta_i)$; for the communication subsystem, the* objective function *$f_{n+1}(X) = G(\lambda)$; $X = [\beta_1, \ldots, \beta_n, \lambda]^T$. The goal is to minimize simultaneously all $f_i(X), i = 1, \ldots, n+1$.*
- *For each player i, $i = 1, \ldots, n+1$, the initial performance $u_i^0 = f_i(X^0)$, where X^0 is a zero vector of length $n+1$.*

Remark 4.1 *In the above definition, we can assume that $\beta_i \leq \hat{\mu}_i$ to satisfy the compactness requirement for X where $\hat{\mu}_i = \mu_i - \epsilon$ for a small $\epsilon > 0$. For simplicity we ignore this in the sequel. We also assume that all the players in the above game are able to achieve performance strictly superior to their initial performance.*

***Theorem* 4.1** *For the cooperative load balancing game defined above there is a unique bargaining point and the bargaining solutions are determined by solving the following optimization problem:*

$$\min_{X} \left[G(\lambda) \prod_{i=1}^{n} D_i(\beta_i)\right] \tag{14}$$

subject to the constraints (11) - (13).

Proof: In the Appendix. □

***Theorem* 4.2** *For the cooperative load balancing game defined above the bargaining solution is determined by solving the following optimization problem:*

$$\min_{X} [\sum_{i=1}^{n} \ln D_i(\beta_i) + \ln G(\lambda)] \quad (15)$$

subject to the constraints (11) - (13).

Proof: In the Appendix. □

***Theorem* 4.3** *The solution to the optimization problem in Theorem 4.2 satisfies the relations*

$$\begin{array}{ll} d_i(\beta_i) \geq \alpha + g(\lambda), & \beta_i = 0 \quad (i \in R_d), \\ d_i(\beta_i) = \alpha + g(\lambda), & 0 < \beta_i < \phi_i \quad (i \in R_a), \\ \alpha + g(\lambda) \geq d_i(\beta_i) \geq \alpha, & \beta_i = \phi_i \quad (i \in N), \\ d_i(\beta_i) = \alpha, & \beta_i > \phi_i \quad (i \in S), \end{array} \quad (16)$$

subject to the total flow constraint,

$$\sum_{i \in S} d_i^{-1}(\alpha) + \sum_{i \in N} \phi_i + \sum_{i \in R_a} d_i^{-1}(\alpha + g(\lambda)) = \Phi \quad (17)$$

where α is the Lagrange multiplier.

Proof: In the Appendix. □

Since obtaining a closed form solution for α from eq. (17) is not possible, we use a simple method such as a binary search to solve eq. (17) iteratively for α as in [7]. Also, from Theorem 4.3, the following properties which are true in the optimal solution can be derived and their proofs are similar to those in [7].

***Property* 4.1**

$$\begin{array}{c} d_i(0) \geq \alpha + g(\lambda), \quad \text{iff} \quad \beta_i = 0, \\ d_i(\phi_i) > \alpha + g(\lambda) > d_i(0), \quad \text{iff} \quad 0 < \beta_i < \phi_i, \\ \alpha \leq d_i(\phi_i) \leq \alpha + g(\lambda), \quad \text{iff} \quad \beta_i = \phi_i, \\ \alpha > d_i(\phi_i), \quad \text{iff} \quad \beta_i > \phi_i. \end{array}$$

***Property* 4.2** *If β is an optimal solution to the problem in Theorem 4.2 then we have*

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

***Property* 4.3** *If β is an optimal solution to the problem in Theorem 4.2 then we have $\lambda = \lambda_S = \lambda_R$, where*

$$\begin{array}{c} \lambda_S = \sum_{i \in S} [d_i^{-1}(\alpha) - \phi_i], \\ \lambda_R = \sum_{i \in R_d} \phi_i + \sum_{i \in R_a} [\phi_i - d_i^{-1}(\alpha + g(\lambda_S))]. \end{array}$$

Based on the above properties, we can have the following definitions (eqs. (18) - (23)) for an arbitrary α (≥ 0).

$$S(\alpha) = \{i | d_i(\phi_i) < \alpha\} \quad (18)$$

$$\lambda_S(\alpha) = \sum_{i \in S(\alpha)} [d_i^{-1}(\alpha) - \phi_i] \quad (19)$$

$$R_d(\alpha) = \{i | d_i(0) \geq \alpha + g(\lambda_S(\alpha))\} \quad (20)$$

$$R_a(\alpha) = \{i | d_i(\phi_i) > \alpha + g(\lambda_S(\alpha)) > d_i(0)\} \quad (21)$$

$$\lambda_R(\alpha) = \sum_{i \in R_d(\alpha)} \phi_i + \sum_{i \in R_a(\alpha)} [\phi_i - d_i^{-1}(\alpha + g(\lambda_S(\alpha)))] \quad (22)$$

$$N(\alpha) = \{i | \alpha \leq d_i(\phi_i) \leq \alpha + g(\lambda_S(\alpha))\} \quad (23)$$

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

$$R_d = R_d(\alpha), R_a = R_a(\alpha), N = N(\alpha), S = S(\alpha) \quad (24)$$

and

$$\lambda = \lambda_S = \lambda_R = \lambda_S(\alpha) = \lambda_R(\alpha) \quad (25)$$

We now present an algorithm (CCOOP) for obtaining the Nash Bargaining Solution for our cooperative load balancing game.

CCOOP algorithm:

Input:

Node processing rates: $\mu_1, \mu_2, \ldots \mu_n$;
Node job arrival rates: $\phi_1, \phi_2, \ldots \phi_n$;
Mean communication time: t.

Output:

Load allocation to the nodes: $\beta_1, \beta_2, \ldots \beta_n$.

1. *Initialization:*
   $\beta_i \leftarrow \phi_i$; $i \in N$; $i = 1, \ldots, n$
2. Sort the computers such that
   $d_1(\phi_1) \leq d_2(\phi_2) \leq \ldots d_n(\phi_n)$
   If $d_1(\phi_1) + g(0) \geq d_n(\phi_n)$, STOP
   (No load balancing is required)
3. Determine α (using a binary search):
   $a \leftarrow d_1(\phi_1)$
   $b \leftarrow d_n(\phi_n)$
   while(1) **do**
   $\lambda_S(\alpha) \leftarrow 0$
   $\lambda_R(\alpha) \leftarrow 0$
   $\alpha \leftarrow \frac{a+b}{2}$
   Calculate: $S(\alpha), \lambda_S(\alpha), R_d(\alpha), R_a(\alpha)$, and $\lambda_R(\alpha)$ (eqs. (18) - (22)) in the order given for $i = 1, \ldots, n$
   If $(|\lambda_S(\alpha) - \lambda_R(\alpha)| < \epsilon)$ **EXIT**
   If $(\lambda_S(\alpha) > \lambda_R(\alpha))$
   $b \leftarrow \alpha$
   else
   $a \leftarrow \alpha$
4. Determine the loads on the computers:
   $\beta_i \leftarrow 0$, for $i \in R_d(\alpha)$
   $\beta_i \leftarrow d_i^{-1}(\alpha + g(\lambda))$, for $i \in R_a(\alpha)$
   $\beta_i \leftarrow d_i^{-1}(\alpha)$, for $i \in S(\alpha)$

$$\beta_i \leftarrow \phi_i, \quad \text{for } i \in N(\alpha) \text{ (eq. (23))}$$

Remark 4.2 *(i) In step 2, we STOP when the total (node + communication) time for a job to be transferred from a more powerful to a less powerful node exceeds the node time on the less powerful node, if the network traffic equals 0. This means that a job will run faster on the 'origin' node than if transferred to a different node. (ii) The running time of this algorithm is $O(n \log n + n \log 1/\epsilon)$, where ϵ denotes the acceptable tolerance used for computing α in step 3 of the algorithm. (iii) This algorithm must be run periodically when the system parameters change in order to recompute a new load allocation.*

5. Experimental Results

5.1. Simulation Environment

We developed a simulation platform to evaluate the performance of our CCOOP scheme. The performance metrics used in our simulations are the *expected response time* and the *fairness index*. The *fairness index* [5], is used to quantify the fairness of load balancing schemes. We perform simulations to study the impact of system utilization and heterogeneity on the performance of the proposed scheme. We also implemented the Overall Optimal Scheme (OPTIM) [7] and the Proportional Scheme (PROP) [1] for comparison. In the following we present and discuss the simulation results.

5.2. Performance Evaluation

Effect of System Utilization. *System utilization* (ρ) represents the amount of load on the system and is defined as the ratio of the total arrival rate to the aggregate processing rate of the system:

$$\rho = \frac{\Phi}{\sum_{i=1}^{n} \mu_i} \tag{26}$$

We simulated a heterogeneous system consisting of 16 computers to study the effect of system utilization. The system has computers with four different processing rates. The system configuration is shown in Table 1.

For each experiment the total job arrival rate in the system Φ is determined by the system utilization ρ and the aggregate processing rate of the system. We choose fixed values for the system utilization and determined the total job arrival rate Φ. The job arrival rate for each computer $\phi_i, i = 1, \ldots, 16$ is determined from the total arrival rate as $\phi_i = q_i \Phi$, where the fractions q_i are given in Table 2. The mean communication time t is assumed to be 0.001 sec.

Table 1. System configuration

Relative processing rate	1	2	5	10
Number of computers	6	5	3	2
Processing rate (jobs/sec)	10	20	50	100

Table 2. Job arrival fractions q_i for each computer

Computer	1-2	3-6	7-11	12-14	15-16
q_i	0.01	0.02	0.04	0.1	0.2

In Figure 2, we present the expected response time of the system for different values of system utilization ranging from 10% to 90%. It can be seen that CCOOP performs as well as OPTIM for ρ ranging from 10% to 40% and is better than PROP for ρ ranging from 50% to 60%. CCOOP approaches PROP at high system utilization.

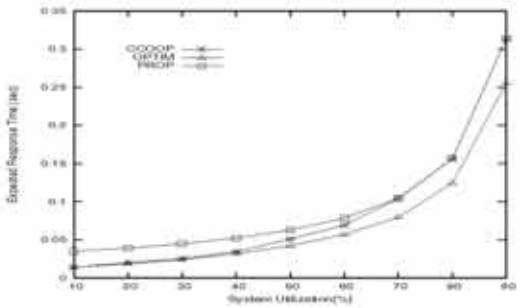

Figure 2. System Utilization vs Expected Response Time

In Figure 3, we present the fairness index for different values of system utilization. The CCOOP scheme has a fairness index of almost 1 for any system utilization. The fairness index of OPTIM drops from 1 at low load to 0.89 at high load and PROP maintains a fairness index of 0.73 over the whole range of system loads.

Effect of Heterogeneity. In this section, we study the effect of heterogeneity on the performance of load balancing schemes. One of the common measures of heterogeneity is the *speed skewness* [16]. We study the effectiveness of load balancing schemes by varying the speed skewness.

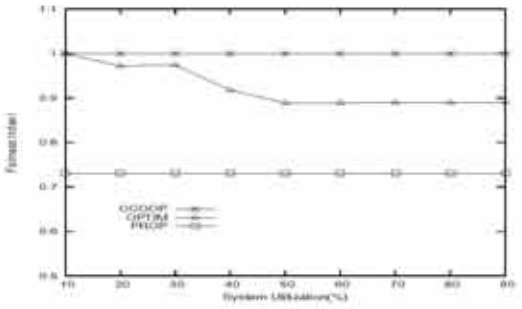

Figure 3. System Utilization vs Fairness Index

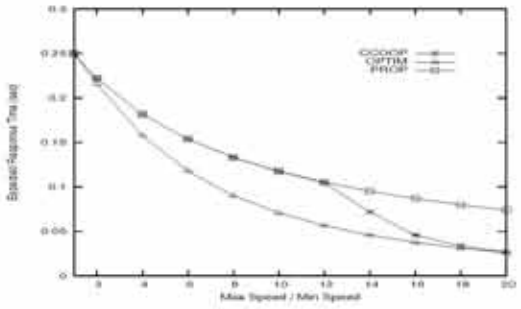

Figure 4. Heterogeneity vs Expected Response Time

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

Figure 4 shows the effect of speed skewness on the expected response time. For low skewness, CCOOP behaves like the PROP. But, as the skewness increases, the performance of CCOOP approaches to that of OPTIM which means that in highly heterogeneous systems CCOOP is very effective.

Fig 5 shows the effect of speed skewness on the fairness index. It can be observed that CCOOP has a fairness index of almost 1 over all range of speed skewness. The fairness index of OPTIM and PROP falls from 1 at low skewness to 0.92 and 0.88 respectively at high skewness.

Table 3. System parameters

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

6. Conclusion

In this paper we presented a game theoretic approach to solve the static load balancing problem in

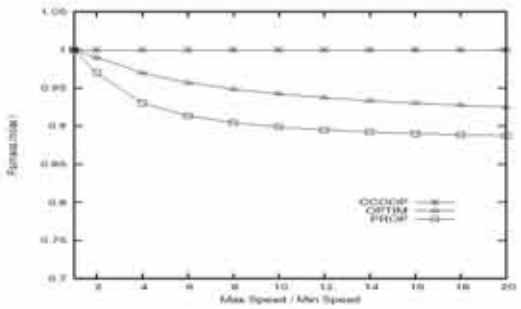

Figure 5. Heterogeneity vs Fairness Index

distributed systems where the computers are connected by a single channel communication network. We used a cooperative game to model the load balancing problem. Our solution is based on the NBS which provides a Pareto optimal and fair solution to the distributed system. For the proposed cooperative load balancing game we derived an algorithm for computing the NBS. The performance of our scheme is compared with that of other existing schemes under simulations. We showed that our scheme is not only fair but also is comparable with that of OPTIM in terms of the mean response time.

Acknowledgements

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

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

Appendix

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

Proof of Theorem 4.1

The objective function for each player $f_i(X)$ (Definition 4.1) is convex and bounded below. The set of constraints is convex and compact. Thus, the conditions in Theorem 2.1 are satisfied and the result follows. □

Proof of Theorem 4.2

The objective vector function $\{f_j\}$, $j \in 1, \ldots, n+1$ (Definition 4.1) of the players is a one-to-one function of X. Thus, applying Theorem 2.1 the result follows.□

Proof of Theorem 4.3

Let u_i and v_i denote the network traffic into node i and the network traffic out of node i respectively. From the balance of the total traffic in the network, we have

$$\sum_{i=1}^{n} u_i = \sum_{i=1}^{n} v_i \tag{27}$$

The load β_i on node i can then be written as

$$\beta_i = \phi_i + u_i - v_i \tag{28}$$

and the network traffic λ can be written as

$$\lambda = \sum_{i=1}^{n} v_i \quad (= \sum_{i=1}^{n} u_i) \tag{29}$$

Hence, the problem becomes:

$$\min E(u,v) = [\sum_{i=1}^{n} \ln D_i(\phi_i + u_i - v_i) + \ln G(\sum_{i=1}^{n} v_i)] \tag{30}$$

subject to

$$\beta_i = \phi_i + u_i - v_i \geq 0, \quad i = 1, \ldots, n \tag{31}$$

$$-\sum_{i=1}^{n} u_i + \sum_{i=1}^{n} v_i = 0 \tag{32}$$

$$\beta_i = \phi_i + u_i - v_i < \mu_i, \quad i = 1, \ldots, n \tag{33}$$

$$u_i \geq 0, \quad i = 1, \ldots, n \tag{34}$$

$$v_i \geq 0, \quad i = 1, \ldots, n \tag{35}$$

The objective function in eq. (30) is convex and the constraints are all linear and they define a convex polyhedron. This imply that the first-order Kuhn-Tucker conditions are necessary and sufficient for optimality [10].

Let $\alpha, \delta_i \leq 0, \eta_i \leq 0, \psi_i \leq 0$ denote the Lagrange multipliers [10] The Lagrangian is

$$L(u,v,\alpha,\delta,\eta,\psi) = E(u,v) + \alpha(-\sum_{i=1}^{n} u_i + \sum_{i=1}^{n} v_i)$$

$$+ \sum_{i=1}^{n} \delta_i(\phi_i + u_i - v_i) + \sum_{i=1}^{n} \eta_i u_i + \sum_{i=1}^{n} \psi_i v_i \tag{36}$$

Remark *We ignore the constraint in eq. (33) since all the associated multipliers will be zero if we introduce it in the lagrangian.*

The optimal solution satisfies the following Kuhn-Tucker conditions:

$$\frac{\partial L}{\partial u_i} = d_i(\phi_i + u_i - v_i) - \alpha + \delta_i + \eta_i = 0$$

$$i = 1, \ldots, n \tag{37}$$

$$\frac{\partial L}{\partial v_i} = -d_i(\phi_i + u_i - v_i) + g(\sum_{i=1}^{n} v_i) + \alpha$$

$$-\delta_i + \psi_i = 0, \quad i = 1, \ldots, n \tag{38}$$

$$\frac{\partial L}{\partial \alpha} = -\sum_{i=1}^{n} u_i + \sum_{i=1}^{n} v_i = 0 \tag{39}$$

$$\phi_i + u_i - v_i \geq 0, \quad \delta_i(\phi_i + u_i - v_i) = 0,$$

$$\delta_i \leq 0, \quad i = 1, \ldots, n \tag{40}$$

$$u_i \geq 0, \quad \eta_i u_i = 0, \quad \eta_i \leq 0, \quad i = 1, \ldots, n \tag{41}$$

$$v_i \geq 0, \quad \psi_i v_i = 0, \quad \psi_i \leq 0, \quad i = 1, \ldots, n \tag{42}$$

In the following, we find an equivalent form of eqs. (37) - (42) in terms of β_i. By adding eqs. (37) and (38) we have, $-g(\sum v_i) = \eta_i + \psi_i$, $i = 1, \ldots, n$. Since $g > 0$, either $\eta_i < 0$ or $\psi_i < 0$ (or both). Hence, from eqs. (41) and (42), for each i, either $u_i = 0$ or $v_i = 0$ (or both). We consider each case separately.

- *Case I:* $u_i = 0, v_i = 0$: Then, we have $\beta_i = \phi_i$. It follows from eq. (40) that $\delta_i = 0$. Substituting this into eqs. (37) and (38) gives

$$d_i(\beta_i) = \alpha - \eta_i \geq \alpha \tag{43}$$

$$d_i(\beta_i) = \alpha + g(\lambda) + \psi_i \leq \alpha + g(\lambda) \tag{44}$$

From the above, we have

$$\alpha \leq d_i(\beta_i) \leq \alpha + g(\lambda) \tag{45}$$

This case corresponds to neutral nodes.

- *Case II:* $u_i = 0, v_i > 0$: Then, from eq. (42), we have $\psi_i = 0$. We consider the following subcases.

  - *Case II.1* $v_i < \phi_i$: Then, we have $0 < \beta_i < \phi_i$. It follows from eq. (40) that $\delta_i = 0$. Substituting this into eqs. (37) and (38) gives

$$d_i(\beta_i) = \alpha - \eta_i \geq \alpha \tag{46}$$

$$d_i(\beta_i) = g(\lambda) + \alpha \tag{47}$$

This case corresponds to active sources.

– *Case II.2* $v_i = \phi_i$: Then, we have $\beta_i = 0$ and eqs. (37) and (38) gives

$$d_i(\beta_i) = \alpha - \delta_i - \eta_i \geq \alpha \quad (48)$$

$$d_i(\beta_i) = \alpha + g(\lambda) - \delta_i \geq \alpha + g(\lambda) \quad (49)$$

Thus, we have

$$d_i(\beta_i) \geq \alpha + g(\lambda) \quad (50)$$

This case corresponds to idle sources.

- *Case III:* $u_i > 0, v_i = 0$: Then, we have $\beta_i > \phi_i$. It follows from eqs. (40) and (41) that $\delta_i = 0$ and $\eta_i = 0$. Substituting this into eq. (37), we have

$$d_i(\beta_i) = \alpha \quad (51)$$

This case corresponds to sinks.

Eq. (39) may be written in terms of β_i as

$$\sum_{i=1}^{n} \beta_i = \Phi \quad (52)$$

Using eqs. (47) and (51), the above equation becomes

$$\sum_{i \in S} d_i^{-1}(\alpha) + \sum_{i \in N} \phi_i + \sum_{i \in R_a} d_i^{-1}(\alpha + g(\lambda)) = \Phi \quad (53)$$

which is the total flow constraint. □

References

[1] 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.

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

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

[4] D. Grosu, A. T. Chronopoulos, and M. Y. Leung. Load balancing in distributed systems: An approach using cooperative games. In *Proc. of the 16th IEEE Intl. Parallel and Distributed Processing Symp.*, pages 52–61, Ft Lauderdale, Florida, USA, April 2002.

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

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

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

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

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

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

[11] A. Mas-Collel, M. D. Whinston, and J. R. Green. *Microeconomic Theory.* Oxford Univ. Press, New York, 1995.

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

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

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

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

[16] 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.

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

[18] 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.

Biographies

Satish Penmatsa received his B.Tech. in Computer Science from Andhra University, India in 2000 and a M.Sc. in Computer Science from the University of Texas at San Antonio in 2003. He is currently pursuing a Ph.D. degree in Computer Science at the University of Texas at San Antonio. His research interests include parallel and distributed systems, grid computing and game theory. He is a student member of the IEEE.

Anthony T. Chronopoulos received his Ph.D. at the University of Illinois in Urbana-Champaign in 1987. He is a senior member of IEEE (since 1997). He has published 39 journal and 51 refereed conference proceedings publications in the areas of Distributed Systems, High Performance Computing and Applications. He has been awarded 13 federal/state government research grants. His work is cited in over 220 non-co-authors' research articles.