THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY

DISTRIBUTED ALGORITHM FOR THE AVERAGE LOAD OF A MULTICOMPUTER

Amnon Barak[†] and Zvi Drezner

CRL-TR-17-84

MARCH 1984

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

[†]On leave from The Hebrew University of Jerusalem, Israel.

engn
UMR0204

DISTRIBUTED ALGORITHMS FOR THE AVERAGE LOAD OF A MULTICOMPUTER

Amnon Barak and Zvi Drezner

March 1984

SUMMARY

In this paper we investigate distributed algorithms for the average load of a multicomputer which consists of a cluster of independent computers that are interconnected by a local area communication network. These algorithms are useful when it is desired to balance the load between the nodes of the multicomputer.

The algorithms that we develop are based on exchange of load messages between the nodes of the multicomputer. Two methods for routing load information are suggested. In the first method, each node sends load messages to a fixed set of prespecified nodes. The second method is based on messages which are sent by each node to a randomly selected node. Based on these information exchanges, we show that each node can find the expected global average of the multicomputer. For each case we give the message exchange algorithm, followed by a proof of convergence to the expected average load.

1. INTRODUCTION

The advent of computer and communication systems has brought with it the opportunities to design multicomputer distributed systems for greater resource sharing, improved performance and reliability. A unique property of distributed operating systems is the possibility for automatic allocation of processor resources to programs, similarly to the allocation of primary memory in traditional operating systems. Automatic allocation allows user programs to be unaware of the amount of resources currently available, and it allows a consistent policy to be applied in order to optimize the allocation throughout the system.

As pointed out by Krueger and Finkel [5], load-balancing is a policy for allocating processor resources. The objective is to assign processes to nodes of the multicomputer in such a way that each node has approximately the same workload. When a process is created, a non-preemptive load balancing algorithm permanently assign it to what appears at that moment to be the best node. The process is not moved even though its run-time characteristics later changes in such a way as to cause the nodes to become very unbalanced. Preemptive load balancing algorithms allow load balancing to occur whenever anomalies appear in the work loads of the nodes. If, in the course of execution, the work loads of the nodes become unbalanced, the process can be migrated to a better node to continue its execution. Load balancing can occur at any time, rather than being limited to times when new processes are created. The increased adaptability of preemptive load balancing makes it more effective than non-preemptive load balancing in a multicomputer having multiprogrammed nodes, particularly if a large number of nodes are used [5].

The objective of a preemptive load balancing algorithm is to balance, or reduce the variance between the loads experienced by nodes of the multicomputer. One way to make the migration decisions correctly, is by finding the average load over all the nodes in the multicomputer, then to migrate processes from nodes whose load are above the average into nodes whose load are below the average. In [5] a method was suggested to determine the global average load by storing it centrally, and making it globally accessible by broadcasting updates of its value every unit of time. It was further suggested that the algorithm could be performed in a distributed manner by allowing any node to broadcast an updated value of the average load whenever it believes that the current value is inaccurate or at a regular time intervals [5]. The main drawback of this method is the broadcast mechanism which it uses. Assuming n node multicomputer. Then in addition to the time required to find the average load, every node is preempted $O(n)$ times during each unit of time, or each time that an update of the average load is broadcasted. These preemptions cause heavy overhead because they require a context switch by every node, for each broadcasted message.

An alternative algorithm for load balancing which uses on the average, only one preemption for each node, during each unit of time, was developed in [1]. In this case, load balancing is achieved by reduction of the variance of loads between pair of nodes of the multicomputer. This algorithm does not need the average load but requires that each node maintains up-to-dated information about the load of a small subset of other nodes. A disadvantage of this algorithm is the communication overhead which results from possible suboptimal process migration between pairs of nodes whose respective loads are below or above the global average.

An optimal algorithm for the average load is computationally expensive because it requires an instant reading of the load of all the nodes. This however can be done only by broadcasts, with a substantial overhead. Alternatively, a suboptimal algorithm can be devised according to the observed state of the system. Such algorithm attempts to approach optimality at a fraction of the cost with less knowledge than is required by an optimal algorithm.

In this paper we propose a class of algorithms by which each node can find an estimate for the global average load. These algorithms are based on exchanges of load information between pairs of nodes, using only $O(n)$ messages during each unit of time. Two methods for message routing are used. In the first method each node sends messages to a fixed, prespecified set of nodes. In section 3 we give algorithms which use this method. In the second method, each node sends load messages to a randomly selected node. An algorithm which uses this method is given in section 4. In each case, we prove that if the load of the nodes remain unchanged then the variance of the estimates for the global average load is reduced during each unit of time. We also give the rate of this reduction.

2. THE MULTICOMPUTER ENVIRONMENT

Consider a multicomputer system having n independent and homogeneous nodes that are interconnected by a local area communication network. Suppose that the topology of the network allows a direct link between any pair of nodes, as for example in a token ring based network or Ethernet. Although the algorithms that we develop

rely on one way communication messages, we assume that each message is instantly acknowledged. This is necessary if the algorithms are to adapt to a dynamic configuration to which nodes may be added or removed.

In a previous paper [1], a preemptive load balancing policy for this environment was developed. This policy consists of the following parts:

1. A load algorithm for monitoring the local processor load.
2. An exchange algorithm which provides each node with information about its load, relatively to other nodes or the global average load.
3. The load balancing algorithm which uses the information provided by the exchange algorithm (and other local optimization considerations) to match between pairs of nodes for possible process migration.
4. The process migration mechanism which physically move processes from one node to another.

In this paper we develop exchange algorithms for finding the global average load. The other parts of the load balancing policy will not be discussed except the mechanism for measuring the processor load which is the basis for any load balancing algorithm. In [1,5] a processor load was defined as the average length of the ready-to-run process queue over a unit period of time t, where t is on the order of time required to migrate an averaged size process. We note that t is a parameter of the system with a typical value of 1 second, as reported in [1].

For the proposed algorithms we assume that each node maintains a small, fixed size load vector. The first component of this vector includes the local processor load

and an estimate of the global average load by that node. The remaining components hold load values and other information about other processors. These vectors are exchanged between the nodes in a manner which provides up-to-date load information to each node.

The algorithms that we develop are intended for a dynamic configuration. This means that if a node sends a message to another node and that node does not acknowledge within a required period of time, then the message is sent to the next, prespecified node in the line. Algorithms which uses this routing method are called deterministic. An alternative method for routing load information to support a dynamic configuration, is to allow a random path by which each node selects a target node at random. For each of the algorithms that we develop we assume that:

1. The number of nodes in the multicomputer is arbitrarily large.
2. Computers may leave or join the network at any time.
3. All the nodes use the same algorithm and the same unit of time t.
4. There is no synchronization between the nodes, and each node is responsible to execute the algorithm independently of the other nodes.

We further assume that each node supports multiprogramming and may work as an independent processor. New jobs may arrive at any node and jobs execute equally well on any node, independent of the node where they arrive. We also assume that no *a priori* information about the execution time of the processes is available.

3. DETERMINISTIC LOAD AVERAGING ALGORITHMS

In this section we develop algorithms for the estimation of the global average load of an n node multicomputer, using a deterministic routing of load information.

Let T be the (absolute) time, as for example seen by an outside observer. Let $L_i(T)$ be the load of node i at time T. Let $L(T)$ be the global average load of the system at time T. Then:

$$L(T) = \frac{1}{n} \sum_{i=1}^{n} L_i(T).$$

Let $A_i(T)$ be the estimate for $L(T)$ by node i, $1 \leq i \leq n$ at time T. Let $A(T)$, and $V(T)$ be the average and variance of the estimates $A_i(T)$ respectively. First, we consider the case in which each node sends its information to one prespecified node. Note that this is equivalent to a ring routing, since the nodes could always be renumbered such that node i sends its load information to node $i+1$, $i = 1, 2, ..., n$. Throughout this paper all the subscripts are calculated *mod n*.

Let $A_{i-1}(T)$ be the load estimate received by node i during the last unit of time t. Note: when node i joins the network then it notifies its predecessor and sets the estimate A_i of the global load to L_i, its local load at that time. Then each node updates its estimate every t seconds using the following steps:

ALGORITHM 1

Step 1: Compute the new local load $L_i(T + t)$.

Step 2: Find the new estimate:

$$A_i(T+t) = \frac{A_i(T) + A_{i-1}(T)}{2} + L_i(T+t) - L_i(T) .$$

Step 3: Send $A_i(T+t)$ to node $i+1$. If that node is not available send to node $i+2$, etc.

Step 4: After t seconds, return to step 1.

Note: Each node does not have the value of T. However, in spite of the lack of synchronization, we assume that all the nodes use the same time increment t to update their corresponding clocks. To simplify the notation of time related functions, when we write

$$f(T+t) = F[f(t)]$$

for any functions f and F, we shall mean the relation between the value of f at the current unit of time, as a (recursive) function of the value of f in the previous unit of time. In these cases the actual value of T is irrelevant. It is important to note that between time T and $T+t$ every node performs the algorithm exactly once. Also, during the interval $(T, T+t)$, each node gets exactly one message from its predecessor.

Theorem 1: Let $L(T)$ be the global average load of the system. If $A(T) = L(T)$ then $A(T+t) = L(T+t)$.

Proof: Assume that during the last unit of time t, the configuration of the multicomputer didn't change, i.e., no new node joined or left the network. Since $A(T) = L(T)$ then:

$$A(T+t) = \frac{1}{n} \sum_{i=1}^{n} A_i(T+t)$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left[\frac{A_i(T) + A_{i-1}(T)}{2} + L_i(T+t) - L_i(T) \right]$$

$$= \frac{1}{2} A(T) + \frac{1}{2} A(T) + L(T+t) - L(T)$$

$$= L(T+t) + A(T) - L(T) = L(T+t) .$$

Note that if one or more nodes joined (or left) the network during the last unit of time, then a straight forward procedure is used to preserve the estimate for the global average load. Also note that due to the accumulation of roundoff errors, it is recommended that from time to time all the nodes restart the algorithm by setting the global load equal to their local load.

As a measure for the effectiveness of Algorithm 1, we find the variance of the A_i's under **stable conditions**, i.e., all the $L_i(T)$ remain constant and there is no change in the configuration.

Let $\vec{A}_k(T) = (A_k(T), A_{k+1}(T), \cdots, A_n(T), A_1(T), \cdots, A_{k-1}(T))$. Let $\rho(T)$ be the correlation coefficient between the vectors $\vec{A}_1(T)$ and $\vec{A}_2(T)$. Note that $-1 \leq \rho(T) \leq 1$.

Theorem 2: Under stable conditions, the variance of the estimates $A_i(T)$ reduces after t seconds by $[1 + \rho(T)]/2$, i.e.,

$$V(T+t) = \frac{1+\rho(T)}{2} V(T) .$$

Proof: By Theorem 1, $A(T+t) = A(T)$. Therefore,

$$V(T+t) = \frac{1}{n}\sum_{i=1}^{n}\left(A_i(T+t) - A(T)\right)^2$$

$$= \frac{1}{4n}\sum_{i=1}^{n}\left(A_i(T) - A(T) + A_{i-1}(T) - A(T)\right)^2$$

$$= \frac{1}{4n}\left\{\sum_{i=1}^{n}\left(A_i(T) - A(T)\right)^2 + \sum_{i=1}^{n}\left(A_{i-1}(T) - A(T)\right)^2\right\}$$

$$+ \frac{1}{2n}\left\{\sum_{i=1}^{n}\left(A_i(T) - A(T)\right)\left(A_{i-1}(T) - A(T)\right)\right\}$$

$$= \frac{1}{4}\left(V(T) + V(T) + 2V(T)\,\rho(T)\right)$$

$$= \frac{1+\rho(T)}{2}\,V(T).$$

The purpose of the following analysis is to prove that under stable conditions the estimates $A_i(T)$ converge to the global average $L(T)$ when $T \to \infty$.

Lemma 1: $V(T+t) < V(T)$, unless $V(T) = 0$.

Proof: We prove that $\rho(T) < 1$.

Assume that $\rho(T) = 1$. Then the vectors $\vec{A}_1(T)$ and $\vec{A}_2(T)$ must be linearly dependent i.e., $A_i(T) = \alpha\, A_{i-1}(T) + \beta$, where $\alpha > 0$ and β are constants. By recursive expansion:

$$A_{n+i}(T) = \alpha\,(\,\alpha\,(\,\cdots\alpha\,A_i(T) + \beta\,) + \beta\cdots\,) + \beta\,.$$

But $A_{n+i}(T) = A_i(T)$, therefore:

$$A_i(T) = \alpha^n A_i(T) + \beta\,(\,1 + \alpha + \cdots + \alpha^{n-1}\,)\,. \qquad (3.1)$$

One solution of (3.1) is $\alpha = 1$. This solution implies $\beta = 0$. Consequently, all the $A_i(T)$ are equal to each other and $V(T) = 0$. This contradicts the assumption of the lemma, therefore we assume $\alpha \neq 1$.

In this case:

$$(1 - \alpha^n)\, A_i(T) = \beta \frac{1 - \alpha^n}{1 - \alpha} .$$

When $\alpha^n = 1$, either $\alpha = 1$, or $\alpha = -1$ for even n. $\alpha = -1$ yields $\rho(T) = -1$, and $\alpha = 1$ was already considered. Therefore, $\alpha^n \neq 1$, and $(1 - \alpha)\, A_i(T) = \beta$, or:

$$A_i(T) = \alpha\, A_i(T) + \beta = A_{i+1}(T),$$

and $V(T) = 0$ in this case too.

In conclusion, $\rho(T) = 1$ and $V(T) > 0$ leads to a contradiction, thus $\rho(T) < 1$.

Lemma 2: For each n there exist $\gamma_n < 1$ such that $V(T + t) \leq \gamma_n V(T)$.

Proof:

$$\gamma_n = sup \frac{V(T+t)}{V(T)} = sup \frac{1 + \rho(T)}{2}$$

$$= \frac{1}{2} + \frac{1}{2} sup \frac{\sum_{i=1}^{n} [A_i(T) - A(T)]\, [A_{i+1}(T) - A(T)]}{V(T)}$$

Let $x_i = A_i(T) - A(T)$.

Then subject to $\sum_{i=1}^{n} x_i = 0$,

$$2\,\gamma_n - 1 = \sup \frac{\sum_{i=1}^{n} x_i\, x_{i+1}}{\sum_{i=1}^{n} x_i^2}\,.$$

Now,

$$\sum_{i=1}^{n} x_i\, x_{i+1} = \frac{1}{2}\left(\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_{i+1}^2 - \sum_{i=1}^{n} (\, x_i - x_{i+1}\,)^2 \right)$$

$$= \sum_{i=1}^{n} x_i^2 - \frac{1}{2} \sum_{i=1}^{n} (\, x_i - x_{i+1})^2.$$

Therefore,

$$\gamma_n = 1 - \frac{1}{4}\ \inf \frac{\sum_{i=1}^{n} (\, x_i - x_{i+1}\,)^2}{\sum_{i=1}^{n} x_i^2}\,, \tag{3.2}$$

subject to: $\sum_{i=1}^{n} x_i = 0$.

In order to solve (3.2) we consider the expression $\sum_{i=1}^{n} (\, x_i - x_{i+1}\,)^2 = x^T M x$, where M is given by:

$$M = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & & & 0 \\ 0 & -1 & 2 & \ddots & & \cdot \\ \cdot & 0 & -1 & \ddots & & \cdot \\ \cdot & & \cdot & \cdots & \cdot & \cdot \\ \cdot & & & \ddots & -1 & 0 \\ 0 & & & \ddots & 2 & -1 \\ -1 & 0 & 0 & \ddots & -1 & 2 \end{pmatrix}$$

Thus (3.2) is equivalent to finding

$$inf \frac{x^T M x}{x^T x} \tag{3.3}$$

subject to: $x^T e = 0$, where $e = (1, 1, ..., 1)$.

M is a symmetric semi-definite matrix, therefore all its eigenvalues are non-negative. In [4] it was shown that the minimal value of $x^T Mx \,/\, x^T x$ is the least eigenvalue of M. It is easy to verify that zero is an eigenvalue of M with an associated eigenvector e. However, this solution does not satisfy the constraint $x^T e = 0$. Now, by [4] the best value for (3.3) when x is orthogonal to the eigenvector associated with the least eigenvalue, is the second least eigenvalue of M. Let us call this eigenvalue λ_2. Therefore, $\gamma_n = 1 - \frac{1}{4}\lambda_2$. By direct calculation, see Appendix 1, we proved that $\lambda_2 = 4\sin^2 \pi/n$, thus $\gamma_n = \cos^2 \pi/n \approx 1 - \pi^2/n^2$.

Theorem 3: Under stable conditions, $\lim_{T\to\infty} A_i(T) = L(T)$.

Proof: By Lemma 2 $\lim_{T\to\infty} V(T) = 0$. By Theorem 1 $\lim_{T\to\infty} A(T) = L(T)$ and the theorem follows.

To assess the behavior of Algorithm 1 for a ring routing, we find the behavior of $V(T)$ under stable conditions, when the $A_i(T)$ are drawn from a random distribution. We also assume that n is large, thus $\rho(T) = 0$, and for $2 \leq k < n$ the correlation coefficient between the vectors $\vec{A}_1(T)$ and $\vec{A}_k(T)$ is zero.

By Theorem 2, $V(T + t) = \frac{1}{2} V(T)$. But this relation does not hold for $V(T + kt)$ for $k \geq 2$ because $\rho(T + kt)$ is no longer zero. To find the variance $V(T + kT)$, for

$k < n$, we first find $A_i(T + kt)$.

$$A_i(T + t) = \frac{1}{2} A_i(T) + \frac{1}{2} A_{i-1}(T);$$

$$A_i(T + 2t) = \frac{1}{4} A_i(T) + \frac{1}{2} A_{i-1}(T) + \frac{1}{4} A_{i-2}(T);$$

In general:

$$A_i(T + kt) = \frac{1}{2^k}\left[\sum_{j=0}^{k}\binom{k}{j} A_{i-j}(T)\right] .$$

Now,

$$V(T + kt) = cov\left\{\frac{1}{2^k}\sum_{j=0}^{k}\binom{k}{j} A_{i-j}(T),\ \frac{1}{2^k}\sum_{j=0}^{k}\binom{k}{j} A_{i-j}(T)\right\}$$

$$= \frac{1}{4^k}\sum_{j=0}^{k}\binom{k}{j}^2 V(T) .$$

The last result follows because $cov\left(A_{i-j_1}(T), A_{i-j_2}(T)\right) = 0$, for $j_1 \neq j_2$.

Define $\alpha_k = \frac{1}{4^k}\sum_{j=0}^{k}\binom{k}{j}^2$. To find α_k, consider the product of the binomial expansion of $(1 + x)^k(1 + \frac{1}{x})^k$. The coefficient of x^0 is $\sum_{j=0}^{k}\binom{k}{j}^2$. On the other hand $(1 + x)^k(1 + \frac{1}{x})^k = \frac{(1 + x)^{2k}}{x^k}$, thus the coefficient of x^0 is $\binom{2k}{k}$. Therefore, $\alpha_k = \frac{1}{4^k}\binom{2k}{k}$, which yields for $k < n$,

$$V(T + kt) = \frac{1}{4^k}\binom{2k}{k}\, V(T)\,.$$

An approximation for α_k is:

$$\alpha_k = \frac{1}{4^k}\frac{(2k)!}{k!^2} \approx \frac{\sqrt{4\pi k}\left(\frac{2k}{e}\right)^{2k}}{4^k 2\pi k\left(\frac{k}{e}\right)^{2k}} = \frac{1}{\sqrt{\pi k}}.$$

Also,

$$\frac{\alpha_k}{\alpha_{k-1}} = \frac{4^{k-1}(2k)!\,[(k-1)!]^2}{4^k(k!)^2\,(2k-2)!} = \frac{2k\,(2k-1)}{4k^2} = \frac{2k-1}{2k}\,.$$

Therefore, for $k < n$.

$$V(T + kt) = \frac{2k-1}{2k}\, V(T + (k\text{-}1)t)\,. \tag{3.4}$$

In conclusion, under stable conditions when using a ring routing the variance is reduced in the k^{th} consecutive step by a factor of $1 - 1/2k$. This yields for k steps, a combined reduction to approximately $1/\sqrt{\pi k}$ of the original variance. We note that this calculations assume that the $L_i(T)$ are constant in time. By comparing (3.4) with Theorem 2 we get that $\rho(T + kt) = (k-1)/k$ for $k < n$. However, if the $L_i(T)$ change in time, variability is added to the $A_i(T)$'s. This is expected to yield a lower $\rho(T + kt)$ thus by Theorem 2, a lower $V(t + kt)$.

3.1 Other Deterministic Algorithms

A drawback of the ring routing is that a message from $L_i(T)$ is transferred to all the other nodes in $O(nt)$ seconds. If we allow each node to exchange more than one

message every unit of time, then it is possible to reduce this "propagation" delay. For example, if we use a two directional ring such that each node sends load messages to its two neighbors during each unit of time, then the propagation delay is reduced by a half. Note however that this scheme uses twice as many messages.

Other possibilities include dense trivalent (cubic) graphs in which each node exchange messages with three neighbors during each unit of time. In this cases the time delay necessary to transmit a message from one node to the other nodes is equal to the diameter of the graph. The diameter is defined as the maximal distance between any pair of nodes, where the distance is the number of nodes in the minimal path connecting the nodes. For example, the family of graphs that are discussed in [6] has a diameter of $3/2 \log n + O(1)$ (where log denotes the base 2 logarithm), and the Cube Connected Cycle [7] diameter is $5/2 \log n + O(1)$.

We now give an algorithm for the general case, when node i receives m estimates, $A_{i_1}, A_{i_2}, \ldots, A_{i_m}$ from nodes $i_1, i_2, \ldots, i_m$ respectively, $0 \leq m \leq n$.

ALGORITHM 2

Step 1: Compute the new local load $L_i(T + t)$.

Step 2: Find the new estimate:

$$A_i(T + t) = \frac{A_i + A_{i_1} + \cdots + A_{i_m}}{m+1} + L_i(T + t) - L_i(T) .$$

Step 3: Send $A_i(T + t)$ to a given number of nodes using a prespecified routing.

Step 4: After t seconds, return to step 1.

Let $p_0, p_1, \ldots, p_n$ be the probabilities that a node gets $0, 1, \ldots, n$ messages respectively during a given unit of time t. Let $E(X)$ denote the expected value of X.

Theorem 4: If $A(T) = L(T)$ then $E(A(T+t)) = L(T+t)$.

Proof:

$$E(A_i(T+t)) = \sum_{k=0}^{n}\left[\frac{A_i(T)\,p_k}{k+1} + \frac{k\,A(T)\,p_k}{k+1}\right] + L_i(T+t) - L_i(T)\,.$$

$$E(A(T+t)) = E\left[\frac{1}{n}\sum_{i=1}^{n} A_i(T+t)\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{n}\left[\frac{A_i(T)\,p_k}{k+1} + \frac{k\,A(T)\,p_k}{k+1}\right] + L(T+t) - L(T)$$

$$= \sum_{k=0}^{n}\frac{p_k}{k+1}\sum_{i=1}^{n}\frac{1}{n}\left[A_i(T) + k\,A(T)\right] + L(T+t) - L(T)$$

$$= \sum_{k=0}^{n}\frac{p_k}{k+1}\left[A(T) + k\,A(T)\right] + L(T+t) - L(T)$$

$$= \sum_{k=0}^{n} p_k\,A(T) + L(T+t) - L(T)\,.$$

Since $\sum_{k=0}^{n} p_k = 1$, we have

$$E(A(T+t)) = A(T) + L(T+t) - L(T) = L(T+t)\,.$$

Theorem 5: The expected variance under stable conditions when the $A_i(T)$ are randomly distributed is:

$$E(V(T+t)) = \sum_{k=0}^{n} \frac{p_k}{k+1} V(T) . \tag{3.5}$$

Proof: Since the $A_i(T)$ are randomly distributed we have:

$$E(V(T+t)) = \sum_{k=0}^{n} p_k E\left[V\left[\frac{A_i(T) + \sum_{j=1}^{k} A_{i_j}(T)}{k+1} \right] \right]$$

$$= \sum_{k=0}^{n} p_k \frac{V(T) + kV(T)}{(k+1)^2} = \sum_{k=o}^{n} \frac{p_k}{k+1} V(T) .$$

Note that Theorem 5 holds only for the first iteration. Then from the second iteration on the $A_i(T)$'s are no longer randomly distributed, thus an analysis must be performed for each specific interconnection network. It is expected that a relationship similar to (3.4) will result.

4. A NONDETERMINISTIC LOAD AVERAGING ALGORITHM

In this section we develop an algorithm for the estimation of the global average load of an n node multicomputer, using a random routing of load information.

ALGORITHM 3

Step 1: Compute the new local load $L_i(T + t)$.

Step 2: Suppose that node i receives m estimates, $A_{i_1}, A_{i_2}, \ldots, A_{i_m}$ from nodes $i_1, i_2, \ldots, i_m$ respectively, $0 \leq m \leq n$. Then find the new estimate:

$$A_i(T + t) = \frac{A_i + A_{i_1} + \cdots + A_{i_m}}{m+1} + L_i(T + t) - L_i(T).$$

Step 3: Choose a random integer j, $j \neq i$ such that $1 \leq j \leq n$.

Step 4: Send the current estimate $A_i(T + t)$ to node j.

Step 5: After t seconds, return to step 1.

Let $p_0, p_1, \ldots, p_n$ be the probabilities that a node gets $0, 1, \ldots, n$ messages respectively during a given unit of time t. Theorems 4 and 5 hold also for Algorithm 3. Next, we find the rate of the reduction of the variance.

Theorem 6: Under stable conditions, Algorithm 3 with random values of $A_i(T)$ satisfies:

$$E(V(T + t)) \approx (1 - 1/e)\, V(T).$$

Proof: Due to the random routing between the nodes, it is easy to verify that the probabilities p_k's are binomially distributed, i.e., $p_k = \binom{n}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k}$.

To calculate the sum $\sum_{k=0}^{n} \frac{p_k}{k+1}$ consider the expression

$$n\int_0^{\frac{1}{n}} (x + 1 - \frac{1}{n})^n \, dx. \tag{4.1}$$

On one hand (4.1) is equal to:

$$\int_0^{\frac{1}{n}} \sum_{k=0}^{n} \binom{n}{k} \, n \, x^k (1 - \frac{1}{n})^{n-k} \, dx = \sum_{k=0}^{n} \frac{p_k}{k+1} .$$

By direct calculation of (4.1),

$$\sum_{k=0}^{n} \frac{p_k}{k+1} = \frac{n}{n+1} \left[1 - (1 - \frac{1}{n})^{n+1} \right] .$$

For large values of n, this expression is approximately $1 - 1/e \approx 0.632$. Note that this ratio is maintained for each successive step because the $A_i(T)$ are randomly distributed.

5. CONCLUSIONS

In this paper we introduced algorithms by which each node can find an estimate for the global average load of a multicomputer distributed system. These algorithms are based on exchange of load messages between the nodes. We analyzed two alternative methods for routing these messages. In the first method, each node sends messages to a set of m neighboring nodes, where $m \geq 1$. We gave discrete algorithms for scattering the load information and proved that for the case $m = 1$, i.e., a ring topology and under stable conditions, i.e., no changes in the configuration, the variance of the estimates of the global average load which is known to each node reduces with the time. We also gave an upper bound for the rate of the reduction of the variance. The second method is based on random scattering of load information. We showed that, under stable conditions, the variance for the estimates of the global average load is reduced by $(1 - 1/e)$ during each unit of time.

In a recent paper [3], we proved that with probability tending to one, the rate of propagation of information from one node to all the other nodes using a random scattering is $D = (1 + \ln 2) \log_2 n$. This suggests that after D units of time, any change in the load of one node will propagate to all the other nodes and that the system will reach equilibrium in a few more steps.

APPENDIX 1

In this appendix we find all the eigenvalues of the matrix M, and in particular λ_2.

Let f be:

$$f = \min \sum_{i=1}^{n} (x_i - x_{i+1})^2 , \tag{A.1}$$

subject to:

$$\sum_{i=1}^{n} x_i^2 = 1 , \tag{A.2}$$

and

$$\sum_{i=1}^{n} x_i = 0 . \tag{A.3}$$

Recall that $x_{n+i} = x_i$.

The Lagrange function for the optimization of (A.1) - (A.3) is:

$$F(X, \lambda, \mu) = \sum_{i=1}^{n} (x_i - x_{i+1})^2 + \lambda(1 - \sum_{i=1}^{n} x_i^2) + \mu \sum_{i=1}^{n} x_i .$$

At a local optimum:

$$\frac{\partial F}{\partial x_i} = 4x_i - 2x_{i-1} - 2x_{i+1} - 2\lambda x_i + \mu = 0. \tag{A.4}$$

Adding (A.4) for all i and recalling (A.3) leads to $\mu = 0$. Substituting in (A.4) yields:

$$(2 - \lambda)x_i = x_{i-1} + x_{i+1}. \tag{A.5}$$

Multiplying (A.5) by x_i and summing for all i gives:

$$(2 - \lambda) \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i x_{i-1} + \sum_{i=1}^{n} x_i x_{i+1} = 2 \sum_{i=1}^{n} x_i x_{i+1}$$

$$= \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} x_{i+1}^2 - \sum_{i=1}^{n} (x_i - x_{i+1})^2 .$$

By (A.2) $2 - \lambda = 2 - f$. Substituting in (A.5) gives:

$$x_{i+1} = (2 - f)x_i - x_{i-1} . \tag{A.6}$$

The solution of (A.6) is:

$$x_i = A\alpha^i + B\beta^i \tag{A.7}$$

where α and β are the roots of the quadratic equation $x^2 = (2 - f)x - 1$, and A, B are constants. Note that these roots are complex numbers:

$$\alpha, \beta = \frac{2 - f \pm i\sqrt{f(4 - f)}}{2} . \tag{A.8}$$

Now, since $x_{n+i} = x_i$ it follows by (A.7) that $\alpha^n = \beta^n = 1$. In particular, α is the n^{th} root of 1, which is for $0 \leq k \leq n - 1$,

$$\alpha = \cos\frac{2\pi k}{n} + i \sin\frac{2\pi k}{n} . \tag{A.9}$$

By comparing (A.8) with (A.9) we get $(2 - f)/2 = \cos 2\pi k/n$ or:

$$f = 2(1 - \cos\frac{2\pi k}{n}) = 4 \sin^2\frac{\pi k}{n,} \qquad \text{for} \quad 0 \leq k \leq n - 1 . \tag{A.10}$$

These are the eigenvalues of M. In particular λ_2 is obtained for $k = 1$ (or $k = n - 1$):

$$\lambda_2 = 4 \sin^2\frac{\pi}{n}$$

UNIVERSITY OF MICHIGAN
3 9015 02229 0137

REFERENCES

[1] Barak A. and Shiloh A., A distributed load balancing policy for a multicomputer, Department of Computer Science, The Hebrew University of Jerusalem, (1984).

[2] Bryant R.M. and Finkel R.A., A stable distributed scheduling algorithm, Proc. Second Int. Conf. Distributed Computing Systems, April 1981.

[3] Drezner Z. and Barak A., A probabilistic algorithm for scattering information in a multicomputer system, Computing Research Laboratory TR-15-84, The University of Michigan, Ann Arbor, Michigan, March 1984.

[4] Hesteness M.R., Optimization Theory - The Finite Dimensional Case, John Wiley & Sons, N.Y. 1975.

[5] Krueger P. and Finkel R.A., An adaptive load balancing algorithm for a multicomputer, Computer Science Department, University of Wisconsin, Madison Wisconsin, 1983.

[6] Leland W.E. and Solomon M.H., Dense Trivalent Graphs for Processor Interconnection, IEEE Trans. Computers, C-31, (1982), 219-222.

[7] Preparata F.P. and Vuillemin J., The Cube-Connected Cycle: A Versatile Network for Parallel Computation, Comm. ACM, 24, (1981), 300-309.