JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 7, 279–301 (1989)

Dynamic Load Balancing for Distributed Memory Multiprocessors*

GEORGE CYBENKO†

Department of Computer Science, Tufts University, Medford, Massachusetts 02155

Received February 4, 1987

In this paper we study diffusion schemes for dynamic load balancing on message passing multiprocessor networks. One of the main results concerns conditions under which these dynamic schemes converge and their rates of convergence for arbitrary topologies. These results use the eigenstructure of the iteration matrices that arise in dynamic load balancing. We completely analyze the hypercube network by explicitly computing the eigenstructure of its node adjacency matrix. Using a realistic model of interprocessor communications, we show that a diffusion approach to load balancing on a hypercube multiprocessor is inferior to another approach which we call the dimension exchange method. For a d-dimensional hypercube, we compute the rate of convergence to a uniform work distribution and show that after $d + 1$ iterations of a diffusion type approach, we can guarantee that the work distribution is approximately within e^{-2} of the uniform distribution independent of the hypercube dimension d. Both static and dynamic random models of work distribution are studied.

1. INTRODUCTION

One of the key issues in algorithm design and problem partitioning for distributed computing is that of load balancing. The goal of load balancing is for each processor to perform an equitable share of the total work load. In many applications, such as dense linear systems solving, it is possible to make a priori estimates of work distribution so that a programmer can build load balancing right into a specific applications program. Such an off-line a priori determination can naturally be called *static* load balancing. By contrast, our

* Some of this work was done at ALPHATECH, Inc., Burlington, MA while the author was on sabbatic leave from Tufts. This research was partially supported by National Science Foundation Grant CCR-87410750, U.S.-Spain Joint Committee for Scientific and Technical Cooperation Grant 84-019, and Office of Naval Research Grant N00014-87-K-01821.

† Present affiliation: center for Supercomputing Research and Development, University of Illinois, Urbana, IL 61801.

0743-7315/89 $3.00

interest here is in situations where no a priori estimates of load distribution are possible. It is only during actual program execution that it becomes apparent how much work is being assigned to individual processors. This can occur in applications such as partial differential equations solvers using adaptively generated grids [1, 2], and data fusion and tracking problems [3]. In such problems, as the computation evolves different processors end up being responsible for differing amounts of work. We shall refer to any strategy for balancing work during an execution as *dynamic*.

This paper restricts attention to distributed memory multiprocessor networks. Data are shared through explicit message passing. One can think of hypercube multiprocessors as prototypical examples of this architecture and indeed the main example in this paper is that of a binary hypercube [4] although many of the results we obtain are for general interconnection topologies.

There has been considerable interest in load balancing in recent years, particularly since the introduction of large scale commercial multiprocessors. Unfortunately the simplest form of load balancing, in which tasks requiring differing times for completion are to be as equally distributed as possible between two processors, is clearly equivalent to the partition problem. A variety of other multiprocessor optimal scheduling problems are NP-Complete as well [5]. As a result of this fact, research on dynamic load balancing has focused on suboptimal procedures that use local information in a distributed memory architecture. Generally speaking, these procedures describe rules for migrating tasks on overutilized processors to underutilized processors in the network. Tradeoffs exist between achieving the goal of completely balancing the load and the communications costs associated with migrating tasks. Approaches that use such a paradigm include [6–9].

We quantify work in terms of *tasks*. All tasks require an equal amount of computational work (that is, time) to be completed and they are indecomposable into subtasks—a single task must be performed by a single processor only. We assume that all tasks are independent and therefore the order in which they are done is irrelevant. Another aspect of this independence is that it does not matter which processor in the network executes which task. Furthermore, tasks are self-contained and can be viewed as segments of data in memory. In order to relocate a task from one processor to another, it is only necessary to communicate the data associated with that task. Of course, in order to make dynamic load balancing an issue, we must consider problems in which the cost of moving a task (as measured in terms of communication delay and time) is significantly smaller than the ultimate time cost of executing the task in place. We have encountered such problems in data fusion and tracking applications where a task is associated with a data record—each data record is repeatedly updated using new information and these data records are independent of one another [10, 3]. Data records are

constantly being added and removed and the distribution of data records is essentially equivalent to the distribution of work.

In spite of the existence of applications where such a situation does arise, this model has some limitations for general applicability. For example, in dynamic grid generation problems, it is important to maintain some locality of tasks (which might correspond to grid points in an irregular grid) to one another. Discussions of load balancing strategies for spatially oriented tasks on regular multiprocessor networks such as hypercubes and grids can be found in [11, 2, 12]. Nonetheless, we believe that the model of this paper is suitable for at least a preliminary study of the general problem.

The model and results of this paper are of interest for a number of reasons. First of all, the load balancing protocols described for general distributed memory networks are extremely simple in comparison with previously studied methods. Second, the general method we study is completely analyzable with respect to its performance and convergence properties. This second fact is noteworthy because most other load balancing schemes have not been analytically studied and in spite of being based on convincing heuristics, their behavior under even simple load evolution has not been completely understood.

This paper is organized as follows. In Section 2 we present the diffusion scheme for general interconnection networks. Necessary and sufficient conditions for the convergence of these schemes on a static work distribution are identified. The results for a static distribution are then used to derive the variance of the work load from a uniform load distribution when work is being created according to a simple stochastic model. Section 3 applies the results of Section 2 to the important case of a binary hypercube multiprocessor. In particular, the optimal homogeneous diffusion rule is obtained for a d-dimensional hypercube. In Section 4 a time dependent scheme for the hypercube is introduced and analyzed. It is shown that the time dependent scheme has better behavior than the general diffusion scheme at least for hypercubes. Section 5 is a summary together with some observations and discussions of unresolved problems.

2. A General Dynamic Load Balancing Scheme

Assume that we have a distributed memory multiprocessor network with n processors labeled 1 through n. At this point we make no assumption about the topology of the network except that it is connected—that is, there is a path between any two processors in the network. In terms of the node adjacency matrix of the graph representing the network, this means that the matrix is irreducible (note that the matrix is symmetric since we assume that communications channels are bidirectional). We will use $N = (V, E)$ to de-

note the network in standard graph notation—V is the set of processors and E is the edge set (edges are the communications channels between processors).

We quantify the work distribution at time t by an n vector, $w^{(t)}$, where $w_i^{(t)}$ is the number of tasks to be done by processor i at time t. Although the number of tasks to be done by processor i is clearly a nonnegative integer, we will treat them as real quantities. In applications with fine task granularity this is a reasonable approximation.

Assume that at time t we have a work distribution of $w^{(t)}$. Our diffusion model for dynamic load balancing has the form

$$w_i^{(t+1)} = w_i^{(t)} + \sum_j \alpha_{ij}(w_j^{(t)} - w_i^{(t)}) + \eta_i^{(t+1)} - c, \tag{1}$$

where α_{ij} are nonnegative constants. The summation is over all processors j in the network with the convention that $\alpha_{ij} = 0$ if i and j are not connected in the network. The interpretation of this formula hinges on the summation terms: processors i and j compare work loads at time t and the amount

$$\alpha_{ij}(w_j^{(t)} - w_i^{(t)})$$

is exchanged. If this quantity is positive, then precisely this amount of work is transferred from processor j to processor i. Should the quantity be negative, the transference is in the other direction. The term $\eta_i^{(t+1)}$ accounts for new work generated at time t for processor i and we will model this as a stochastic phenomenon shortly. The term c is a constant amount of work that any processor can accomplish between times t and times $t + 1$.

This technique for dynamic load balancing has been informally proposed by a few researchers primarily because of its simplicity and its analogy with the physical process of diffusion (that is, a process governed by a parabolic partial differential equation). It is appealing to think of work diffusing in a natural way through the multiprocessor network. Another interpretation of this approach involves analogies with finite Markov chain models. Work distribution can be considered to be an initial probability distribution and the diffusion of work is mathematically identical to the evolution of state occupation probabilities. Accordingly, it is not surprising that most of the mathematical tools and ideas are in fact derived from both Markov chain theory and the numerical analysis of matrix iterative schemes.

In order to analyze the behavior of (1), we must first study a simpler static model and then use linearity of the basic equations to get results for the general model. Our static model is given by the simplified equations

$$w_i^{(t+1)} = w_i^{(t)} + \sum_j \alpha_{ij}(w_j^{(t)} - w_i^{(t)}) \tag{1a}$$

which are obtained from (1a) by dropping the inhomogeneous terms. Rewriting (1a), we have

$$w_i^{(t+1)} = (1 - \sum_j \alpha_{ij}) w_i^{(t)} + \sum_j \alpha_{ij} w_j^{(t)} \tag{2}$$

which clearly illustrates the need for two fundamental constraints on the constants α_{ij}:

(a) $\alpha_{ij} \geq 0$ for each i and j;

(b) $1 - \sum_j \alpha_{ij} \geq 0$ for every i.

Note that the relationship given by (2) for the dynamic updating of work distribution is linear and we can write a simple vector equation for the update. Specifically, we have that

$$w^{(t+1)} = Mw^{(t)}, \tag{3}$$

where $M = (m_{ij})$ is given by

$$m_{ij} = \begin{cases} \alpha_{ij} & \text{if} \quad i \neq j \\ 1 - \sum_k \alpha_{ik} & \text{if} \quad i = j. \end{cases} \tag{4}$$

Note that

$$\sum_i m_{ij} = 1 \tag{5}$$

and that M is symmetric. In particular, M is a doubly stochastic matrix and virtually all of our analysis depends on this fact. We shall call a matrix M obtained from a dynamic load balancing rule given by the α_{ij} as in (4) a *diffusion* matrix. A final bit of terminology concerns the notion of the *average* work distribution. For a given work distribution w, we let $\bar{w}$ be the vector whose every entry is exactly

$$\sum_i w_i / n = w^* u / n,$$

where u is the n-vector all of whose components are 1 and w^* denotes matrix transposition. This uniform distribution allocates the same amount of work to every processor while keeping the total amount of work in the network constant. Convergence of an iterative dynamic load balancing strategy is thus meant to mean convergence of the network work distribution to the uniform

one. The distance from one distribution to another is given by the Euclidean distance between distributions considered as n-vectors.

The matrix M leaves the average work load invariant as a consequence of property (5) above. For completeness we have:

LEMMA 1. $\bar{w}^{(t)} = \bar{w}^{(0)}$ *for all* $t \geq 0$.

Proof. Expand $Mw^{(0)} = w^{(1)}$, and use (5) to simplify. The result then follows by induction on t. ■

In the following, we use various properties of M to establish conditions under which the iteration (3) converges to the uniform load distribution and the rate at which the convergence takes place. We will use the powerful Perron–Frobenius theory of irreducible nonnegative matrices and we refer the reader to [13] for an exposition and derivation of the results we invoke. In the terminology of [13], observe that M is a symmetric irreducible nonnegative matrix. The Perron–Frobenius theory asserts a number of facts about such a matrix M. For completeness, we state the relevant results.

THEOREM (Perron–Frobenius [13]). *Let M be a positive, irreducible matrix with row sums all equal to* 1. *Then the following are true:*

(a) *the eigenvalues of M of unit magnitude are the kth roots of unity for some k and all are simple;*

(b) *the eigenvalues of M of unit magnitude are kth roots of unity if and only if M is similar under a permutation to a k cyclic matrix;*

(c) *all eigenvalues of M are bounded by* 1.

Lemma 1 actually demonstrates that an eigenvector for 1 is the uniform distribution u and that within scaling this must therefore be the only eigenvector for 1. Now since the eigenvalues of M are bounded by 1, standard arguments [13] show that the iteration (3) converges to the uniform distribution if and only if -1 is not an eigenvalue as well.

Let

$$\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_2 < \lambda_1 = 1$$

be the ordered eigenvalues of M with $\lambda_1 = 1$. Note that the last inequality is strict since 1 is a simple eigenvalue. Let

$$\gamma(M) = \max_{i>1} |\lambda_i|.$$

Our goal is to determine situations where $\gamma < 1$. We first need an auxiliary idea.

DEFINITION. Given a network N and a diffusion matrix M as above, the *induced* network or graph is given by $N_m = (V, E_M)$, where for $i, j \in V$ we have $(i, j) \in E_M$ if and only if $\alpha_{ij} > 0$.

The induced network is essentially the original network with possibly some channels deleted—the channels removed are precisely the ones on which no exchange of work is made, that is, channels for which $\alpha_{ij} = 0$. Using this notion of induced graph, we now state one of our major results.

THEOREM 1. *The iteration* (3) *always converges to the uniform distribution if and only if the induced network is connected and either* (*or both*) *of the following conditions hold:*

(i) $(1 - \sum_i \alpha_{ij}) > 0$ *for some* j;

(ii) *the induced graph is not bipartite.*

Proof. First we note that the requirement of connectedness of the induced network is trivially needed for the uniform distribution to be the only eigenvector of M.

By the Perron–Frobenius theory for irreducible nonnegative matrices, M has eigenvalue -1 precisely when the nodes of the network can be ordered to give M the form

$$M = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}, \tag{6}$$

where 0's are used to denote square zero block matrices on the diagonal of M and A is a rectangular nonnegative matrix. The basic reasoning here is that since M is symmetric, it must be either primitive or cyclic of order 2. Assuming -1 is an eigenvalue, M must then be cyclic of order 2 and hence the cyclic canonical form is given by (6) on account of symmetry. Now suppose that M has the form in (6). Observe that for the eigenvector

$$\begin{bmatrix} u \\ u \end{bmatrix}, \tag{7}$$

where as before u is the vector of 1's (with dimensions compatible to M's partitioning in (6)), we have

$$M\begin{bmatrix} u \\ -u \end{bmatrix} = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}\begin{bmatrix} u \\ -u \end{bmatrix} = (-1)\begin{bmatrix} u \\ -u \end{bmatrix} \tag{8}$$

so that -1 is indeed an eigenvalue of M.

Now M can be permuted into the form (6) if and only if both

$$m_{jj} = 1 - \sum_i \alpha_{ij} = 0 \qquad \text{for all } j \tag{9}$$

and

$$\text{the induced network } N_M \text{ is bipartite.} \tag{10}$$

To see this observe that the diagonal entries of M are invariant as a set under permutations of nodes (simultaneous row and column permutations of M) so (9) is evidently necessary for M to have the canonical form of (6). Futhermore, it is a simple exercise to see that M has the canonical form of (6) if the induced network is bipartite (merely 2-color the nodes using say black and red and put the black nodes first followed by the red nodes in the ordering). On the other hand, if M has the form of (6) then the block structure of M determines a 2-coloring of the induced network and so it is bipartite.

Thus far we have shown that -1 is not an eigenvalue of M, that is, that $\gamma < 1$, if either of the conditions of the theorem are true. The remainder of the proof is a standard convergence argument whose details we forego. We simply state that if γ is defined as above, then

$$\| Mw - \bar{w} \|^2 \leqslant \gamma^2 \| w - \bar{w} \|^2 \tag{11}$$

and so by induction

$$\| M^t w - \bar{w} \|^2 \leqslant \gamma^{2t} \| w - \bar{w} \|^2.$$

Thus the iteration converges if $\gamma < 1$. If $\gamma = 1$, it is easy to construct examples that fail to converge. Specifically, consider

$$v^{(0)} = \begin{bmatrix} \phi u \\ \psi u \end{bmatrix} = \begin{bmatrix} \dfrac{(\phi + \psi)}{2} u \\ \dfrac{(\phi + \psi)}{2} u \end{bmatrix} + \begin{bmatrix} \dfrac{(\phi - \psi)}{2} u \\ \dfrac{(\psi - \phi)}{2} u \end{bmatrix} = v_1 + v_2.$$

Then

$$M^t v^{(0)} = v^{(t)} = v_1 + (-1)^t v_2$$

and for $\phi \neq \psi$ the iteration does not converge. ■

Given a diffusion matrix, there is a class of simple scalings that can transform the matrix to another diffusion matrix. Note that adding a scalar multi-

ple, say κ, of the identity matrix to M merely translates the eigenvalues of M by κ and changes the row and column sums to be $1 + \kappa$. M can then be renormalized by dividing it by $1 + \kappa$. Summarizing, we shall call a matrix of the form

$$M(\kappa) = (M + \kappa I_n)/(1 + \kappa)$$

a *linear* modification of M. Notice that not all linear modifications lead to physically realistic diffusion matrices. Clearly $\kappa > 0$ never presents a problem but $\kappa < 0$ can if some diagonal entries of M are zero. In general, note that

$$\kappa \geqslant -\min_j m_{jj}$$

leads to the only legal linear modifications since we cannot have a diffusion rule that distributes away more work than exists at a node.

There is a simple rule for determining linear modifications that give optimal rates of convergence for the resulting matrices, subject to the feasibility constraint above.

LEMMA 2. *Suppose that M is a diffusion matrix. Then the optimal feasible linear modification is obtained by setting κ equal to the larger of*

$$-(\lambda_2 + \lambda_n)/2, \qquad -\min_j m_{jj},$$

where as before λ_2 is the second largest eigenvalue of M and λ_n is the smallest eigenvalue.

Proof. The eigenvalues of M are of the form

$$\lambda_n, \lambda_{n-1}, \ldots, \lambda_2, 1$$

and the eigenvalues of $M(\kappa)$ are

$$(\lambda_n + \kappa)/(1 + \kappa), \ldots, (\lambda_2 + \kappa)/(1 + \kappa), 1.$$

It is an easy matter to verify that without the feasibility constraint,

$$\min_\kappa \gamma(M(\kappa))$$

is attained for

$$\kappa = -(\lambda_n + \lambda_2)/2$$

and that as a function of κ, $\gamma(M(\kappa))$ is unimodal with a single minimum. If the optimal κ is unfeasible, then we take the smallest legal value for κ. This minimizes $\gamma(M(\kappa))$. By the proof of the previous result, minimizing $\gamma(M)$ gives the fastest convergence rate for the iteration. ■

Theorem 1 and Lemma 2 give a complete analysis of general diffusion schemes for general topologies with a static work distribution. Our assumption in this static model was that the work load was initially determined by $w^{(0)}$ and that no work was created or completed between iterations of the diffusion scheme. Let us now return to the case where new work is created and that a constant amount of work is done by every processor between iterations. Our original model from (1) is recalled here:

$$w^{(j+1)} = Mw^{(j)} - cu + \eta^{(j+1)} \tag{12}$$

with $w^{(0)} = \eta^{(0)}$. The reader is reminded that the interpretation of the two additional terms on the right of (12) is:

—each processor completes exactly c tasks between iterations of the dynamic scheme thereby reducing the amount of work by c units;

—$\eta^{(j+1)}$ is a random vector, each element of which is drawn independently from a probability distribution on the positive reals whose mean is μ and variance is σ^2.

The three cases where $\mu > c$, $\mu = c$, $\mu < c$ correspond to three distinct situations. In the first case, the expected uniform load distribution grows unboundedly as t grows indefinitely while the case $\mu < c$ is the case where the expected work distribution reaches zero eventually. The case of real practical interest is when $\mu = c$ since this means that the expected work distribution is a finite constant distribution throughout time. Our analysis of this model however does not make any explicit assumptions about the size of μ relative to c since we state our results in terms of the expected deviation from the uniform distribution regardless of what it is or how it changes in time.

Note that (12) does not guarantee that every entry of $w^{(j)}$ is positive because of the randomness assumption and the fact that we decrement by a constant amount of work independently of how much work there is pending at a processor. If we modify (12) by allowing only positive entries, the system equation (12) becomes nonlinear and impossible to analyze using these techniques. To deal with this problem we consider the following interpretations:

(a) assume that $w^{(0)} = Cu$, where C is a large positive constant so that with very high probability, the entries of $w^{(j)}$ remain positive throughout the iteration;

(b) use (12) with the negative entries as is and treat it as an approximation to the realistic situation (we can actually think of carrying negative work

loads as the ability of a processor to store up computational power if it is not totally used during one iteration—this may correspond to performing some excess of system overhead that frees future computation for the application being balanced);

(c) let $c = 0$ in cases where the work w quantifies the number of objects that have to be *managed* by the various processors in the network (such as distributed searching or data updating).

In any case, we can proceed with an analysis of the convergence of (12) without specifying what the actual situation is. The statement of our main result for this general model is in terms of variances and expectations. We let E denote expectation with respect to the probability distribution of the entries of $\eta^{(j)}$.

We state a simple result without proof that will be used in the following.

LEMMA 3. *Suppose that $\zeta_1, \ldots, \zeta_n$ are n independent identically distributed random variables with variance σ^2. Then for $1 \leqslant k \leqslant n$*

$$E(|\sum_{i=1}^{k} \zeta_i/k - \sum_{i=1}^{n} \zeta_i/n|^2) = \frac{(n-k)\sigma^2}{nk}.$$

THEOREM 2. *Suppose that M is a diffusion matrix with $\gamma(M) < 1$. Assume* (12) *and that*

$$E(\eta_i^{(j)}) = \mu \quad \text{for} \quad j > 0$$
$$E(\eta_i^{(0)}) = C$$
$$E(|\eta_i^{(j)} - \mu|^2) = \sigma^2 \quad \text{for} \quad j > 0$$
$$E(|\eta_i^{(0)} - C|^2) = \sigma_0^2.$$

Then $E(w^{(j+1)})$ is a uniform work distribution and

$$E(\|w^{(j+1)} - \bar{w}^{(j+1)}\|^2) \leqslant (n-1)\sigma \frac{(1-\gamma^{2j+2})}{(1-\gamma^2)} + (n-1)\sigma_0^2 \gamma^{2j+2}.$$

Proof. Using the fact that $Mu = u$, we can expand (12) as

$$w^{(j+1)} = \sum_{i=0}^{j+1} M^{j-i+1} \eta^{(i)} - (j+1)cu$$

and using the linearity of the expectation operation, E, we have

$$E(w^{(j+1)}) = \sum_{i=0}^{j+1} M^{j-i+1} E(\eta^{(i)}) - (j+1)cu = (C + (j+1)(\mu - c))u$$

which is a uniform distribution of work in the network. Note that

$$\bar{w}^{(t+1)} = \bar{w}^{(t)} + \bar{\eta}^{(t+1)} - cu$$

and that $E(\bar{w}^{(t+1)}) = E(w^{(t+1)})$ by linearity. Thus

$$Mw^{(t)} - \bar{w}^{(t)}$$

and

$$\eta^{(t+1)} - \bar{\eta}^{(t+1)}$$

are zero mean independent variables (since $w^{(t)}$ consists only of random variables independent of $\eta^{(t+1)}$).

Thus we have

$$\begin{aligned} E(\|w^{(j+1)} - \bar{w}^{(j+1)}\|^2) &= E(\|Mw^{(j)} - \bar{w}^{(j)}\|^2) + E(\|\eta^{(t+1)} - \bar{\eta}^{(t+1)}\|^2) \\ &\leqslant \gamma^2 E(\|w^{(j)} - \bar{w}^{(j)}\|^2) + (n-1)\sigma^2 \\ &\leqslant \sum_{i=0}^{j} \gamma^{2i}(n-1)\sigma^2 + \gamma^{2j+2} E(\|\eta^{(0)} - \bar{\eta}^{(0)}\|^2) \\ &= (n-1)\sigma^2 \frac{(1-\gamma^{2j+2})}{(1-\gamma^2)} + (n-1)\sigma_0^2 \gamma^{2j+2}, \end{aligned}$$

where we have used Lemma 3 with $k = 1$. ■

Letting j grow, meaning that we let the computation run indefinitely, we see that the limiting variance is

$$(n-1)\sigma^2/(1-\gamma^2)$$

which can be viewed as the expected steady state deviation from a uniform distribution. By contrast, consider the variance of the load distribution without balancing—the load at a node of the network is simply the sum of the contributions locally over time. So at time t, this sum includes $t+1$ independent random variables with variance σ^2 each and the variance of the sum is the sum of the variances. Thus the variance of the unbalanced distribution is $(t+1)\sigma^2$ at time t in spite of the fact that the expected value of the distribution is uniform. Put another way, although the unbalanced load has an expected value that is uniform, its expected deviation from the uniform distri-

bution is unbounded as time grows indefinitely. This shows that while the variance of an unbalanced distribution would become arbitrary large, the dynamic load balancing scheme controls variance growth and keeps it bounded. Note that as expected, the convergence factor γ is bounded by 1 and the variance is largest when the convergence is slowest.

3. An Analysis of the Binary Hypercube

For a d-dimensional hypercube, let us first consider the dynamic load balancing rule given by

$$\alpha_{ij} = 1/d \qquad \text{if nodes } i \text{ and } j \text{ are connected in the hypercube}$$

and call the resulting diffusion matrix M_d. Let us consider the matrix $C_d = dM_d$ which has entries consisting of zeros and ones. C_d is the node adjacency matrix for a d cube. We have the following basic result about the eigenvalues of C_d.

THEOREM 3. *The eigenvalues of C_d and their multiplicities are given by the generating function*

$$(x + x^{-1})^d \tag{13}$$

in the sense that the multiplicity of an eigenvalue λ of C_d is the coefficient of x^λ in the expansion of (13).

Proof. A standard way to construct a d-dimensional hypercube is to take two $(d-1)$-dimensional hypercubes and connect corresponding nodes. In matrix terms, it is easy to verify that we get

$$C_d = \begin{bmatrix} C_{d-1} & I \\ I & C_{d-1} \end{bmatrix}. \tag{14}$$

Our argument is inductive and we note that for $d = 1$ the eigenvalues of C_1 are 1 and -1. Now assume the result is true for the $(d-1)$-dimensional hypercube.

Let λ be an eigenvalue for C_{d-1} and suppose that z is a corresponding eigenvector. Then we have

$$C_d \begin{bmatrix} z \\ z \end{bmatrix} = (\lambda + 1) \begin{bmatrix} z \\ z \end{bmatrix}$$

and

$$C_d \begin{bmatrix} z \\ -z \end{bmatrix} = (\lambda - 1) \begin{bmatrix} z \\ -z \end{bmatrix}$$

which can be verified directly by using the form of C_d in (14). If z_1 and z_2 are two orthogonal eigenvectors for the same eigenvalue λ it is easy to verify that the resulting

$$\begin{bmatrix} z_1 \\ z_1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} z_2 \\ z_2 \end{bmatrix}$$

are orthogonal as well as are

$$\begin{bmatrix} z_1 \\ -z_1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} z_2 \\ -z_2 \end{bmatrix}.$$

Moreover, if z_1 is an eigenvalue for λ of C_{d-1} and z_2 is an eigenvalue for $\lambda - 2$ then they are orthogonal since C_{d-1} is symmetric and

$$\begin{bmatrix} z_1 \\ -z_1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} z_2 \\ z_2 \end{bmatrix}$$

are eigenvectors for C_d for the eigenvalue $\lambda - 1$ and they are orthogonal as well.

Thus we have seen that the

$$\begin{bmatrix} z \\ z \end{bmatrix}$$

construction produces eigenvalues and multiplicities generated by $(x + x^{-1})^{d-1}x$ while the

$$\begin{bmatrix} z \\ -z \end{bmatrix}$$

construction produces eigenvalues and multiplicities generated by $(x + x^{-1})^{d-1}x^{-1}$. By the above orthogonality arguments, the eigenvalues and multiplicities of C_d are given by the summed generating functions and this establishes the inductive step. ■

By this theorem, the eigenvalues for a d-dimensional hypercube are

$$-d, -d+2, -d+4, \ldots, d-4, d-2, d$$

with multiplicities

$$1, d, d(d-1)/2, \ldots, d(d-1)/2, d, 1,$$

respectively. Thus the eigenvalues of M_d are

$$-1, -1+2/d, \ldots, 1-2/d, 1$$

and by Lemma 2 the optimal linear modification is obtained by letting

$$\gamma = \frac{1-(1-2/d)}{2} = \frac{1}{d}$$

and the resulting convergence factor is

$$\kappa = \frac{1-\gamma}{1+\gamma} = 1 - \frac{2}{(d+1)}.$$

Thus after d iterations of the optimal linearly modified load balancing scheme, we have that

$$\|w^{(d)} - \bar{w}\| \leq \left(1 - \frac{2}{(d+1)}\right)^d \|w^{(0)} - \bar{w}\|. \tag{15}$$

Note that the optimally modified scheme has the following rule for a d-dimensional hypercube: exchange $1/(1+d)$ of the difference in work between neighboring processors.

Furthermore, we note that d iterations lead to the factor in (15) (which is best possible as can be checked)

$$\left(1 - \frac{2}{(d+1)}\right)^d \tag{16}$$

which is asymptotically equal to e^{-2} (about 0.13). This means that d iterations of the optimal diffusion for a d-dimensional hypercube bring the initial

distribution about 8 times closer to the uniform distribution than the initial distribution was independently of d.

In the next section, we will show that under realistic assumptions about communication costs in a hypercube network, there are methods that are guaranteed to give the exact uniform work distribution after d communication steps.

4. A Time Dependent Scheme for the Hypercube

The diffusion schemes for dynamic load balancing discussed thus far have had the property that they essentially acted simultaneously on all interprocessor communications channels. For all current hardware implementations of message passing systems that we know about, this is unrealistic. Returning to the strategy summarized by (2) or (12), let us examine the actual sequence of communication steps required to implement such an algorithm. First of all, processor i must exchange its current work load, $w_i^{(0)}$, with each neighbor before actually transmitting or receiving the appropriate number of tasks from each neighbor. Now although machines such as the INTEL iPSC or NCUBE hypercubes have parallel hardware for communications, the communications must be serialized with respect to an individual processor. That is, it appears to be impossible to simultaneously communicate with all neighbors. Thus it is largely irrelevant whether all work load distribution communications are made first, followed by communication of the actual tasks, or load distribution communications are interleaved with task communications.

With this in mind, suppose a processor communicates with a neighbor and exchanges some amount of work with that neighbor before communicating with another neighbor. By virtue of having already exchanged some amount of work with the first neighbor, it is possible to use the updated work distribution as a basis for subsequent exchanges with other neighbors. This comparison is similar to the differences between Jacobi and Gauss–Seidel methods for iterative matrix methods [13]. We will examine such a scheme for the hypercube interconnect network.

Suppose that we have a d-dimensional binary hypercube. Consider the following pairings of nodes: for $1 \leqslant i \leqslant d$ consider the pairs of processors whose binary labels differ in the ith bit. Each such pair will exactly balance the work distribution between them. That is, each pair ends up with half the total work those two processors have. We repeat this process for each dimension between 1 and d, using at each step the current updated work load at each processor. Let us call this the *dimension exchange* load balancing scheme.

Before proceeding with an analysis of the scheme, notice that under our model assumption that communications have to be serialized with respect to individual processors, one sweep through all dimensions of the hypercube is equivalent to a single iteration of the diffusion approach. Once again, we believe this is the realistic model for current hypercube machines. Even if the communications in the diffusion approach can be parallelized, we see that d iterations of the diffusion method can be done in the same time required by one sweep of the dimension exchange method. In the last section we saw (inequality (15) and the discussion following it) that d iterations of the diffusion approach bring us about 8 times closer to the uniform distribution than was the initial work distribution. This was independent of the dimension d. We will now show that one sweep of the dimension exchange method balances the work distribution exactly for the case of a static work distribution. Furthermore, we analyze the dimension exchange scheme for the dynamically changing work distribution as given by the model in (12).

In order to study the dimension exchange scheme, we need to introduce some notation. For $1 \leq i \leq d$, consider the matrix $D_i = (d_{jk}(i))$, where

$$d_{jk}(i) = \begin{cases} 1 & \text{if } j \text{ and } k \text{ differ in bit } i \text{ only} \\ 0 & \text{otherwise.} \end{cases}$$

In words, D_i describes the adjacency relationship when we restrict attention to the i dimension only. Balancing exactly in the ith dimension as described above is determined by the reducible diffusion matrix

$$F_i = (I + D_i)/2.$$

Thus the dimension exchange scheme is given by

$$w^{(t+1)} = F_{(t+1) \bmod d+1} w^{(t)},$$

where the $(t + 1) \bmod d + 1$ expression merely expresses the fact that we cycle through the d bits. Our first result is that we achieve a uniform distribution after one sweep through the dimensions.

THEOREM 4. *$\prod_{i=1}^{d} F_i = 2^{-d}U$, where U is the matrix whose every entry is 1. Since $2^{-d}Uw = \bar{w}$ for any work distribution w, this means that sweeping through all dimensions yields a uniform distribution. Moreover, the same result holds regardless of the order in which the product is formed.*

Proof. We shall show that

$$\prod_{i=1}^{d} (I + D_i) = U.$$

Expanding the product, notice that it is the sum of terms of the form

$$D_i D_j \cdots D_k$$

over all tuples of distinct dimensions $i, j, \ldots, k$. Such a product matrix has a 1 as its α, β entry if and only if α and β differ in bits $i, j, \ldots, k$ and 0 otherwise. Since the bit positions in which every pair of processors in a hypercube differ are unique and since a single term with those dimensions will appear in the expanded product, we see that the sum is indeed U. The claim about being able to change the order of the factors in the product without affecting the results follows from the invariance of the hypercube under permutations of dimensions. ■

In passing, we note that this result can also be established by induction in a manner similar to that of Theorem 3. That approach would require introducing additional notation however.

We now address the behavior of the dimension exchange method under the model given by (12)—that is, the case where new work is introduced at each step and pending work is completed. Since a complete sweep through all dimensions gives a uniform distribution and since the action of doing one dimension step is linear as determined by the matrix F_i, we need only examine the behavior for d consecutive steps or one complete sweep.

Assume that $\eta^{(t)}$ is a random vector as before and that μu is its mean. The variance of an element of $\eta^{(t)}$ is σ^2. Our time dependent analog of (12) is given by the systems equation

$$w^{(t+1)} = F_{(t+1) \bmod d+1} w^{(t)} + \eta^{(t+1)} - cu. \tag{17}$$

THEOREM 5. *Assuming the model of* (17) *we have*

$$E(\| w^{(t+1)} - \bar{w}^{(t+1)} \|^2) \leqslant n\sigma^2.$$

Proof. Since as already noted this scheme uniformly distributes any work load after d steps, we can assume that $t + 1 = d$ and that $w^{(0)} = \eta^{(0)}$ is the initial random distribution with the given statistics. Then

$$w^{(d)} = \sum_{t=1}^{d+1} \left(\prod_{j=t}^{d} F_j\right) \eta^{(t-1)} - dcu$$

and

$$\bar{w}^{(d)} = \sum_{t=0}^{d} \bar{\eta}^{(t)} - dcu.$$

As before, the variables

$$\prod_{j=t}^{d} F_j \eta^{(t-1)} - \bar{\eta}^{(t-1)}$$

are zero mean independent random variables for $1 \leq t \leq d+1$. We have

$$\begin{aligned} E(\|w^{(d)} - \bar{w}^{(d)}\|^2) &= \sum_{t=1}^{d+1} E(\| \prod_{j=t}^{d} F_j \eta^{(t-1)} - \bar{\eta}^{(t-1)} \|^2) \\ &= \sum_{t=1}^{d+1} 2^d \frac{(2^d - 2^{t-1})}{2^{d+t-1}} \sigma^2 \leq \sum_{t=1}^{d+1} 2^{d-t+1} \sigma^2 \qquad (18) \\ &\leq (2^{d+1} - 1)\sigma^2 \leq 2n\sigma^2, \end{aligned}$$

where the equality (18) requires some explanation. Note that

$$\prod_{i=1}^{j} F_j \eta \qquad (19)$$

can be interpreted as follows: the d-dimensional hypercube is partitioned into 2^{d-j} subcubes of dimension j and the work load on each subcube is balanced exactly. Thus we invoke Lemma 3 on each component of (19) with $n = 2^d$ and $k = 2^j$. There are $n = 2^d$ such components. ■

We can compare the diffusion load balancing scheme with this dimension exchange method on the basis of Theorems 2, 3, 4, and 5. From the discussion following Theorem 3 and Eq. (15) in particular, we see that applying Theorem 2 to the hypercube diffusion method we get an asymptotic variance of

$$(n-1)\sigma^2/(1-\gamma^2) = \frac{(n-1)(d+1)^2}{4d+8}\sigma^2$$

noting that $n = 2^d$ for the hypercube. This is about

$$\frac{1}{4} n \log n \sigma^2 \qquad (20)$$

for large d. Our result for the dimension exchange scheme is that the asymptotic variance is bounded by

$$2n\sigma^2$$

so that based on a statistical, long-term analysis, the dimension exchange approach is better in the sense that it gives a smaller variance of the actual distribution from the corresponding uniform distribution. Recall that this analysis used the assumption that one step of the diffusion approach was equivalent to one step (out of d steps in a complete sweep) of the dimension exchange method which favors the diffusion method considerably. If we assume the more realistic model that one step of diffusion uses the same communication complexity as a complete sweep of the dimension exchange method, then dimension exchange looks even better. Since a complete sweep of dimension exchange uniformly distributes all work, our comparison in this case just pits the variance in (20) with

$$E(\|\eta - \bar{\eta}\|^2) = (n-1)\sigma^2$$

which is better by a logorithmic factor than diffusion.

5. Discussion and Summary

The hypercube results in this paper can be easily summarized using tables for comparing the diffusion method with the dimension exchange approach. The comparisons are made for both models of communications and for both models of work—namely fixed work loads or randomly generated work distributions. We remind the reader that our comparisons are based on upper bounds of convergence factors and variances and allowances must be made for loose bounds. However, a close examination shows that the inequalities used were as tight as possible given the techniques we use. We believe the bounds we derive are adequate for purposes of comparison.

First, let us specify the two communications models that we used. Model A is the realistic model where all communications relative to a processor have to be serialized. Thus for the diffusion approach, this means that we really need $2d$ communication steps for one iteration—d steps to communicate work load information with all neighbors and d steps for doing the actual work balancing with all neighbors. Thus under Model A, we can perform one complete sweep of the dimension exchange method.

Model B assumes that communications with respect to a single processor can be done in parallel so that diffusion requires 2 communications rounds

(one for work load information communication and one for actual work load exchange). On the other hand, dimension exchange would still require $2d$ steps since the results of exchanges in prior dimensions are relevant to future exchanges.

Thus Model A allows us to make only one iteration of diffusion for every complete cycle of the dimension exchange method while Model B permits d iterations of diffusion for one sweep of dimension exchange. Model B clearly favors diffusion but as we have already stated, this model is not realistic for today's hardware implementations of hypercube multiprocessors and besides, dimension exchange is superior according to our analysis even under this model.

Our first table summarizes our findings for fixed initial work distributions—namely those that are considered in Theorems 1 and 4. The quantities, C, in Table I are the constants we have computed in inequalities of the form

$$\|w^* - \bar{w}^*\| \leqslant C\|w - \bar{w}\|,$$

where w^* is the work distribution after making the appropriate number of communications steps as determined by the underlying model.

Next we summarize our findings for the statistical model for work distribution. Referring the reader back to Eqs. (12) and (17), where random vectors of work were being introduced at every time interval, we list the asymptotic variances of the two schemes under the two models of communication (Table II).

It is evident that our analysis has shown that dimension exchange has uniformly better performance as a load balancing strategy for hypercubes than does diffusion.

The analysis carries a few caveats as already discussed. It is appropriate to summarize restricting assumptions we have made in order to carry out our analyses. First of all, we assumed that all tasks were independent and could be done on any processor in any sequence. Second, we assumed that all communications required proportionately the same amounts of time in order to compare Models A and B for diffusion and dimension exchange strategies.

TABLE I

Model	Diffusion	Dimension exchange
A	$1 - \frac{2}{d+1}$	0
B	e^{-2}	0

TABLE II

Model	Diffusion	Dimension exchange
A	$\frac{1}{4} n \log n\sigma^2$	$n\sigma^2$
B	$\frac{1}{4} n \log n\sigma^2$	$2n\sigma^2$

It would be appropriate to assume that communications times were actually proportional to the number of tasks being sent to a neighbor but this appears to be a very difficult problem to model and study. Finally, our analysis of the statistical situation was rather simple, assuming that incoming tasks were identically distributed and independent. Even with all these restrictions and simplifications, we believe that the analysis of this paper is the first rigorous treatment of dynamic load balancing strategies and can serve as a basis for future development and research into the subject. In passing we mention that similar analyses are possible for other concrete networks such as rings and grids but since the hypercube is currently the most promising interconnection network for distributed memory architectures, we restricted our study of specific topologies to that one.

In addition to analyzing other regular interconnection networks (assuming some sort of homogeneous or natural diffusion matrix), there are related open problems. For example, given an arbitrary interconnection network, how can one determine a diffusion scheme that has an optimal convergence rate (that is, a minimal γ as developed in Section 2)? On another tack altogether, there is the question of studying asynchronous diffusion load balancing schemes along the lines of the asynchronous iteration schemes discussed in [14], for example. It can be shown that asynchronous rules converge under quite general conditions (see [15], for example)—however, no effective analysis of the convergence rate appears to be known.

In summary, our study shows that in spite of the appealing qualities of diffusion as a load balancing strategy, the deterministic dimension exchange scheme we have presented has better convergence properties at least for hypercubes. This paper has presented a general approach for studying the convergence rates of diffusion schemes for load balancing and completely characterizes the conditions under which the schemes converge. We have shown how convergence rates are computed from the diffusion matrices and how they can be optimized with a simple class of modifications.

ACKNOWLEDGMENTS

The author thanks David Castanon, Tom Allen, and David Krumme for numerous comments and suggestions that contributed to this work. The anonymous referees made suggestions improving the results presented.

REFERENCES

1. Keyes, D. E., and Gropp, W. D. A comparison of domain decomposition techniques for elliptic partial differential equations and their implementation. *SIAM J. Sci. Statist. Comput.* **8** (1987), 166–202.
2. Berger, M. J., and Bokhari, S. A partitioning strategy for non-uniform problems on multiprocessors. *IEEE Trans. Comput.* **C-26** (1987), 570–580.
3. Cybenko, G., and Allen, T. G. Parallel algorithms for classification and clustering. *Proc. SPIE Conference on Advanced Architectures and Algorithms for Signal Processing.* San Diego, CA, 1987.
4. Heath, M. T. (Ed.). *Hypercube Multiprocessors 1986,* SIAM, Philadelphia, 1986.
5. Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, 1979.
6. Chou, T. C. K., and Abraham, J. A. Load balancing in distributed systems. *IEEE Trans. Software Engrg.* **8** (1982), 401–412.
7. Nicol, D. M., and Reynolds, P. F., Jr. Dynamic remapping decisions in multi-phase parallel computations. *ICASE Tech. Rep.* **86-58** (1986).
8. Lin, F. C. H., and Keller, R. M. The gradient model load balancing method. *IEEE Trans. Software Engrg.* **13** (1987), 32–37.
9. Ni, L. M., Xu, C.-W., and Gendreau, T. B. A distributed drafting algorithm for load balancing. *IEEE Trans. Software Engrg.* **11** (1985), 1153–1161.
10. Allen, T. G., Cybenko, G., Polito, J., and Angelli, C. Hypercube implementation of tracking algorithms. *Proc. JDL Workshop on Command and Control,* Washington, DC, 1987.
11. Baden, S. B. Dynamic load balancing of a vortex calculation running on multiprocessors. Lawrence Berkeley Laboratory Research Report, Vol. 22584, 1986.
12. Allen, T. G., and Cybenko, G. Recursive binary partitions, submitted for publication.
13. Varga, R. S. *Matrix Iterative Analysis.* Prentice–Hall, Englewood Cliffs, NJ, 1962.
14. Lubachevsky, B., and Mitra, D. A chaotic asynchronous algorithm for computing the fixed point of a nonegative matrix of unit spectral radius. *J. Assoc. Comput. Mach.* **33** (1986), 130–150.
15. Bertsekas, D. P., and Tsitsiklis, J. N. *Parallel and Distributed Algorithms.* Prentice–Hall, Englewood Cliffs, NJ, 1989.

GEORGE CYBENKO is Professor of Electrical and Computer Engineering, holding joint appointments at the Center for Supercomputing Research and Development and the Coordinated Sciences Laboratory, all at the University of Illinois at Urbana–Champaign. Professor Cybenko received his B.Sc. in mathematics from the University of Toronto (1974) and his Ph.D. in applied mathematics of electrical engineering and computer science from Princeton University (1978). He is Associate Editor of *Mathematics of Control, Signals and Systems* (Springer-Verlag), the *SIAM Journal on Matrix Analysis and Applications,* and the University of Manchester Press Series on *Advanced Algorithms and Architectures for Scientific Computing.*