

Nearest neighbor algorithms for load balancing in parallel computers *

Chengzhong Xu †

Dept. of Electrical & Computer Engg.
Wayne State University, Detroit, MI 48202
czxu@ece.eng.wayne.edu

Burkhard Monien and Reinhard Lüling ‡

Department of Computer Science
University of Paderborn, Germany
{bm, rl}@uni-paderborn.de

Francis C.M. Lau

Department of Computer Science
The University of Hong Kong
fcmlau@cs.hku.hk

Abstract

With nearest neighbor load balancing algorithms, a processor makes balancing decisions based on localized workload information and manages workload migrations within its neighborhood. This paper compares a couple of fairly well-known nearest neighbor algorithms, *the dimension-exchange* (DE, for short) and *the diffusion* (DF, for short) methods and their several variants—the average dimension-exchange (ADE), the optimally-tuned dimension-exchange (ODE), the local average diffusion (ADF) and the optimally-tuned diffusion (ODF). The measures of interest are their efficiency in driving any initial workload distribution to a uniform distribution and their ability in controlling the growth of the variance among the processors' workloads. The comparison is made with respect to both one-port and all-port communication architectures and in consideration of various implementation strategies including synchronous/asynchronous invocation policies and static/dynamic random workload behaviors. It turns out that the dimension-exchange method outperforms the diffusion method in the one-port communication model. In particular, the ODE algorithm leads itself to best suited for statically synchronous implementations of a load balancing process regardless of its underlying communication models. The strength of the diffusion method is in asynchronous implementations in the all-port communication model; the ODF algorithm performs best in that case. The underlying communication networks considered assume the most popular topologies, the mesh and the torus and their special cases: the hypercube and the k -ary n -cube.

*A version of this paper appeared in *Concurrency, Practice and Experience*, Vol. 7(7), pp. 707-736, 1995.

†Partly supported by NSF MIP-9309489.

‡Partly supported by the DFG Sonderforschungsbereich 376 "Massive Parallelität" and the EC ESPRIT Long Term Research Project 20244 (ALCOM-IT).

1 Introduction

Massively parallel computers have been shown to be very efficient in solving problems that can be partitioned into tasks with static computation and communication patterns. However, there exist a large class of problems that have unpredictable computational requirements or irregular communication patterns. To solve this kind of problems efficiently in parallel computers, it is necessary to perform load balancing operations at run-time.

The execution of a load balancing procedure requires some means of maintaining a global view of the system and some negotiation mechanism for workload migrations across processors to take place. Every load balancing strategy has to resolve the issues of when to invoke a balancing operation, who makes load balancing decisions according to what information, and how to manage workload migrations between processors. Combining different answers to the above yields a large space of possible designs of load balancing algorithms with widely varying characteristics. Nearest neighbor algorithms are such a class of methods in which processors make decisions based on local information in a decentralized manner and manage workload migrations within the immediate neighborhood [1, 2, 3, 4, 5]. Since they would only spread local workload to nearest neighbors, these algorithms can be easily scaled to operate in massively parallel computers of any size, and would tend to preserve the communication locality inherent in the underlying computations. In general, these algorithms are executed iteratively, with the expectation that successive invocations of local load balancing would eventually bring about a global balanced state; hence, they give the flexibility of controlling the balance quality over a spectrum of possibilities, from load sharing (no idle processors coexist with busy processors) to the global balanced state.

Nearest neighbor load balancing algorithms rely on successive approximations to a global uniform distribution, and hence at each operation, need only be concerned with the direction of workload migration and the issue of how to apportion excess workloads. Among existing load balancing methods that are characterized by different choices of the direction of workload migration [6], we are interested in the *diffusion* and the *dimension-exchange* methods. These two methods have drawn a fair amount of attention in recent years. With the diffusion method, a heavily or lightly loaded processor balances its workload with all of its nearest neighbors simultaneously in a load balancing operation [7, 8]. With the the dimension-exchange method, a processor in need of load balancing balances its workload successively with its neighbors one at a time, and each time a new workload index is computed, which will be used in the the subsequent pairwise balancing [8, 5, 9]. These two methods are closely related, and they lend themselves particularly well to implementation in two basic communication architectures, the *all-port* and the *one-port* models, respectively. The all-port model allows a processor to exchange messages with all its direct neighbors simultaneously in one communication step, while the one-port model restricts a processor to exchange messages with at most one direct neighbor at one time. Both of these two models were assumed in many recent researches on communication algorithms [10, 11]. Although the latest designs of message-passing processors tend to support all-port simultaneous communications, the restrictive one-port model is still valid in existing real parallel computer systems. Since the cost in setting up a communication is fixed, the total time spent in sending d messages to d different ports, assuming the best possible overlapping in time, is still largely determined by d unless the messages are rather long.

The all-port and one-port models favor the diffusion and the dimension-exchange methods, respectively. In a system that supports all-port communications, a load balancing operation

using the diffusion method can be completed in one communication step while that using the dimension-exchange method would take d steps. It appears that the diffusion method has an advantage over the dimension-exchange method as far as exploiting the communication bandwidth is concerned. A natural but interesting question is whether the advantage translates into real performance benefits in load balancing or not. The performance of a load balancing algorithm is determined by two measures. One is *efficiency* which is reflected by the number of communication steps required by the algorithm to drive an initial workload distribution into a uniform distribution. This measure alone is sufficient for those kinds of problems that need global balancing at run time. However, for the other kinds of applications that need to achieve load sharing rather than global balancing, we need another measure, the *balance quality*, to reflect the ability of the algorithm in bounding the variance of processors' workloads after performing one or more load balancing operations. The objective of this study is to answer the question concerning the performance of the diffusion and the dimension-exchange methods in different communication models.

In the literature, the diffusion and the dimension-exchange methods have received a lot of attention from both theoretical and experimental researchers. The diffusion method was first modeled using linear system theory by Cybenko [8], and Bertsekas and Tsitsiklis [7]. Cybenko showed that the diffusion method will eventually coerce any initial workload distribution into a global uniform distribution in static situations in which no workloads are generated or consumed during load balancing, and presented an asymptotic bound for the variance of any workload distribution during load balancing in the dynamic situation. Similar convergence results in the static situation were obtained independently by Boillat [12]. Boillat also proved that the diffusion load balancing will converge to a global balanced state in polynomial time. The diffusion method in the dynamic situation was studied by Hong *et al.* [13], and Qian and Yang [14], as well. They presented a constant bound for the variance of workload distribution when applying the method to some specific structures. The diffusion method is characterized by a parameter which determines the portion of excess workload to be diffused away. Xu and Lau analyzed the effects of the parameter on the efficiency of the diffusion method, and derived its optimal values for the mesh and the torus networks [15].

The dimension-exchange method was conceptually designed for hypercube-structured parallel computers, in which balancing proceeds iteratively in dimensions. At each dimension, a processor balances its workload with that of its neighbor belonging to the dimension. Cybenko showed that regardless of the order of dimensions considered, this simple load balancing method yields a uniform distribution from any initial workload distribution after a round of balancing operations [8]. He also revealed the superiority of the dimension-exchange method over the diffusion method in terms of their efficiencies and balance qualities.

The dimension-exchange method is not limited to hypercube structures. Hosseini *et al.* applied this method to arbitrary structures based on edge-coloring [16]. Furthermore, Xu and Lau showed that "equal splitting" of the workload in a pairwise balancing operation might not lead to maximum efficiency in most popular structures, such as the mesh and the torus, although it performs best in the hypercube [5, 9]. Through introducing an exchange parameter to govern the splitting of workload at every step, they derived the optimal values in closed form for the n -D mesh and torus structures.

The theoretical study of the diffusion and the dimension-exchange methods established their sound mathematical foundation. On the practical side, the benefits of the diffusion method were demonstrated in the context of distributed computations of branch-and-bound algorithms [17, 4], and the dimension-exchange method was experimented in parallel graph

partitioning [18] and periodic re-mapping of data parallel computations [19]. Also, Willebeek-LeMair and Reeves [4] compared the results of these two methods in the distributed computation of branch-and-bound algorithms on a hypercube-structured iPSC/2. Their experiments concluded that the speedup due to the dimension-exchange method is better than the speedup due to the diffusion method. It is in agreement with the Cybenko's result.

Although the results of both theoretical and experimental study point to the superiority of the dimension-exchange method in hypercubes, it might not be the case for other popular networks. On the other hand, previous theoretical studies of these two methods were mostly on their *synchronous* implementations in which all processors participate in load balancing operations simultaneously and each processor cannot proceed into the next step until the workload migrations demanded by the current operation have completed. Relatively little results have been obtained on the asynchronous implementations of these methods. Bertsekas and Tsitsiklis proved the convergence property of an asynchronous implementation of the diffusion method [7], and Song extended the result to the case of the total workload being too small to be divided infinitely [20]. Lüling and Monien considered a randomized version of the diffusion method in which a processor in need of load balancing activates an operation among a number of randomly chosen neighbors, and showed that the algorithm will keep the workload difference between any two processors bounded [21]. However, none of these works addressed both the issues of efficiency and balance quality together.

In this paper, we make a comprehensive comparison between the diffusion and the dimension-exchange methods in terms of their efficiency and balancing quality when they are implemented in both one-port and all-port communication models, using synchronous/asynchronous invocation policies, and with static/dynamic random workload behaviors. The communication networks to be considered include the structures of n -D torus and mesh, and their special cases: the ring, the chain, the hypercube and the k -ary n -cube. The mesh and the torus allow different number of nodes in different dimensions. A k -ary n -cube is a special case of the n -D torus in that it has k nodes in each dimension [22, 23]. The hypercube is a special case of both the n -D mesh and the k -ary n -cube. A hypercube is an n -D mesh having two nodes in each dimension, that is, a 2-ary n -cube. We limit our scope to these structures because they are the most popular choices of topologies in commercial parallel computers [23, 24].

Both the dimension-exchange and the diffusion methods are parameterized methods. Their performance is largely influenced by the choice of the parameter values. We focus on two choices of the parameter value in each method: the average dimension-exchange (ADE), the optimally-tuned dimension-exchange (ODE), the local average diffusion (ADF), and the optimally-tuned diffusion (ODF). The optimality here is in terms of the efficiency in static synchronous implementations among various choices of the dimension-exchange and the diffusion parameters. The average versions are the most original versions when the methods were first proposed and are still being employed in real applications today. Our main results are that the dimension-exchange method outperforms the diffusion method in the one-port communication model; in particular, the ODE algorithm is found to be best suited for synchronous implementation in the static situation; and that the dimension-exchange method is superior in synchronous load balancing even under the all-port communication model; the strength of the diffusion method is in asynchronous implementation under the all-port communication model; the ODF algorithm performs best in this case.

The rest of paper is organized as follows. We first present a generic model of load balancing in Section 2, which provides a framework for the comparison of the load balancing algorithms. Section 3 describes the load balancing algorithms in a unified form. In both Sec-

tion 4 and Section 5, the algorithms are compared with respect to their implementation using asynchronous and synchronous invocation policies, respectively. Section 6 gives the results from simulations, which verify our theoretical results as well as provide further information on these load balancing algorithms. We conclude in Section 7 with a summary of the results of the comparison between the dimension-exchange and the diffusion methods.

2 A Generic Model of Load Balancing

We consider a class of parallel computers which are composed of a finite set of homogeneous processors interconnected by a direct communication network. Processors communicate through passing messages. The communication channels are assumed to be full duplex so that a pair of directly connected (nearest neighbor) processors can send/receive messages simultaneously to/from each other. In addition, we assume that the operations of sending and receiving messages through a channel can take place instantaneously. We represent such a system by a simple connected graph $G = (V, E)$, where V is a set of processors labeled 1 through N , and $E \subseteq V \times V$ is a set of edges. Every edge $(i, j) \in E$ corresponds to the communication channel between processors i and j . Let $\mathcal{A}(i)$ denote the set of nearest neighbors of processor i , $d(i) = |\mathcal{A}(i)|$ be the degree of processor i , and d be the maximum of $d(i)$ for $1 \leq i \leq N$.

The underlying parallel computation is assumed to comprise a large number of independent processes, which are the basic units of workload. The total number of processes are assumed to be large enough so that the workload of a processor is infinitely divisible. Processes may be dynamically generated, consumed, or migrated due to imbalance as the computation proceeds. We classify the operations into two types: the computational operation and the balancing operation. At any time, a processor can perform a computational operation, a balancing operation, or both operations simultaneously. The concurrent execution of these two operations is possible when processors are capable of multiprogramming or the balancing operation is done in the background by special coprocessors. The workload of processors can be either fixed or varying with time during the load balancing operation, which we refer to as the *static* and the *dynamic* situations, respectively.

Let t be a time variable, representing global real time. We quantify the workload of processor i at time t by w_i^t in terms of the number of residing processes. We use integer time to simplify the presentation. The results can be expended readily to continuous time. Let $\mathcal{I}(t)$ denote the set of processors performing balancing operations at time t . The change of workload of a processor at time t can be modeled by the following equation in the static situation

$$w_i^{t+1} = \begin{cases} w_i^t + \phi_i^{t+1} & i \notin \mathcal{I}(t) \\ f_i(w_i^t, w_j^t \mid j \in \mathcal{A}(i)) & i \in \mathcal{I}(t) \end{cases} \quad (1)$$

and the following equation in the dynamic situation

$$w_i^{t+1} = \begin{cases} w_i^t + \phi_i^{t+1} & i \notin \mathcal{I}(t) \\ f_i(w_i^t, w_j^t \mid j \in \mathcal{A}(i)) + \phi_i^{t+1} & i \in \mathcal{I}(t) \end{cases} \quad (2)$$

where ϕ_i^{t+1} denotes the amount of workload generated or finished from time t to $t+1$, and $f_i(\cdot)$ represents a load balancing operator.

This model is generic because the load balance operator $f_i(\cdot)$ and the set of processors in load balancing at any time t , $\mathcal{I}(t)$, are left unspecified. The operator $f_i(\cdot)$ can be any nearest neighbor load balancing algorithm, including the diffusion and the dimension-exchange methods. The set $\mathcal{I}(t)$ is determined by the invocation policy of the load balancing. The choice of $\mathcal{I}(t)$ is orthogonal to the load balancing algorithm in that any invocation policy can be used in combination with any load balancing algorithm in implementation. Since a load balancing operation incurs non-negligible overheads, different applications require different invocation policies for a better tradeoff between performance benefits and overheads. In parallel computations using domain decomposition techniques, for example, the computational requirement associated with each portion of a problem domain may change as the computation proceeds. An effective way to reduce the penalty due to load imbalances is to periodically re-decompose the problem domain with the aim of achieving a global uniform distribution across the processors. To this end, all processors are required to perform load balancing operations synchronously for a short time period. That is, $\mathcal{I}(t) = \{1, 2, \dots, N\}$ for $t \geq t_0$, where t_0 is the instance when the global system state satisfies certain conditions such as those set in [25]. By contrast, the parallel execution of dynamic tree-structured computations usually requires only load sharing—assuring that no idle processors exist while there are other busy processors. Thus, each processor is allowed to invoke a load balancing operation at any time in an asynchronous manner according to its own local workload distribution. A simple policy is to activate a load balancing operation once a processor’s workload drops below a preset threshold, $w_{\text{underload}}$, *i.e.*, $\mathcal{I}(t) = \{i | w_i^t < w_{\text{underload}}\}$. More sophisticated invocation policies were discussed in [21, 4]. In short, we make a distinction between *synchronous* and *asynchronous* implementations of load balancing according to their invocation policies. Figure 1 presents one example of these two implementation models in a system of five processors. The dots and the triangles represent the computational operations and the load balancing operations, respectively.

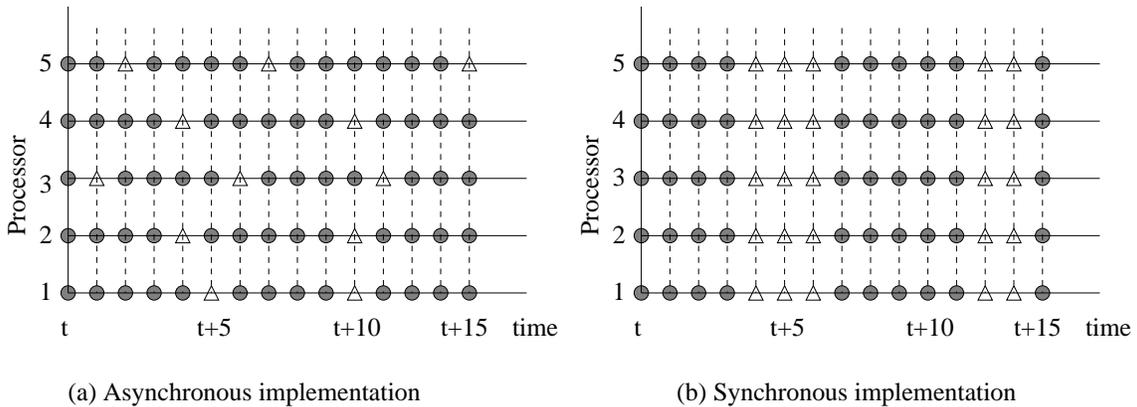


Figure 1: An illustration of generic models of load balancing

Assume $t = 0$ when processors invoke a synchronous or asynchronous load balancing procedure. We are concerned with subsequent workload distributions resulting from different load balancing algorithms. Denote the overall workload distribution at certain time t by a vector $W^t = (w_1^t, w_2^t, \dots, w_N^t)$. Denote its corresponding uniform distribution by a vector $\bar{W}^t = (\bar{w}^t, \bar{w}^t, \dots, \bar{w}^t)$, where $\bar{w}^t = \sum_{i=1}^N w_i^t / N$. We define the workload variance, denoted by

ν^t , as the deviation of W^t from \overline{W}^t ; that is,

$$\nu^t = \|W^t - \overline{W}^t\|^2 = \sum_{i=1}^N (w_i^t - \overline{w}^t)^2.$$

With the workload variance ν^t , we define the efficiency of a load balancing algorithm as the number of load balancing steps required to reduce the variance of the initial state to a tolerable level in the static situation; and define the balance quality as the bound for the variance which is to be guaranteed by the load balancing procedure in the dynamic situation. Load balancing algorithms will be compared in terms of these two measures under the following assumption. Throughout the paper, $E[\cdot]$ denotes the expected value of a random variable.

Assumption 2.1 *Initially, processors' workloads, w_i^0 , $1 \leq i \leq N$, are N independent and identically distributed (i.i.d.) random variables with expectation μ_0 and variance σ_0^2 . At any time t , $t \geq 0$, processors' workload generation/consumption amount, ϕ_i^t , $1 \leq i \leq N$, are zero in the static situation or i.i.d. random variables with expectation μ and variance σ^2 in the dynamic situation.*

3 The Dimension-Exchange and the Diffusion Methods

This section briefly describes the dimension-exchange and the diffusion methods. Both of them are parameterized load balancing algorithms. We present several instances of these two methods based on different choices of values for their parameters.

3.1 The Dimension-Exchange Method

With the dimension-exchange method, any processor which invokes a load balancing operation balances its workload with its neighbors successively. For a processor i , it works in the following way

$$f(\cdot) \equiv \begin{array}{l} \mathbf{for} \quad (c = 1; c \leq d(i); c++) \\ \quad w_i = w_i + \lambda(w_{j_c} - w_i) \end{array} \quad (3)$$

where $j_c \in \mathcal{A}(i)$; and $0 < \lambda < 1$, called the dimension-exchange parameter, is given a fixed value beforehand which determines the fraction of excess workload to be migrated between a pair of processors. The formula says that a balancing operation in the dimension-exchange method comprises $d(i)$ pairwise balancing steps for processor i . At each step, processor i balances its workload with one of its neighbors, and uses the new result for the subsequent balancing. It is because of the sequential nature in the sequence of balancing steps, a load balancing operation requires $d(i)$ communication steps in both the all-port and the one-port communication models.

The efficiency of the dimension-exchange method is determined by the dimension-exchange parameter. A dimension-exchange operation with different choices of the parameter will reduce the workload variance of the system by different degrees. In the following, we present two choices of the parameter which have been suggested as rational choices in the literature.

1. *Average dimension-exchange* (ADE) equally splits the total workload of a pair of processors – that is, $\lambda = 1/2$. It is a straightforward choice for local balancing at each pairwise operation, and has been favored in hypercube-structured systems [8, 26, 27].

2. *Optimally-tuned dimension-exchange* (ODE) takes certain specific parameter values that have the effect of maximizing efficiency in static and synchronous balancing [5, 9]. The optimal parameter depends on the topology and the size of underlying communication network. Let $k = \max\{k_i, 1 \leq i \leq n\}$ in the $k_1 \times k_2 \cdots \times k_n$ mesh and torus. Then, their optimal parameter values were shown, in [9], to be

- $\lambda = 1/(1 + \sin(\pi/k))$ in the mesh,
- $\lambda = 1/(1 + \sin(2\pi/k))$ in the torus.

The dimension-exchange method can be implemented without difficulty in cases where only a few processors that are not close to each other are in need of load balancing at the same time. However, its synchronous implementation requires processors to be coordinated in order to parallelize balancing operations along different communication channels as well as to avoid communication collisions. The potential of parallel efficiency is due to the fact that the execution order of pairwise balancing steps in the operation $f(\cdot)$ of Eq. (2) is left undefined. The parallelization of pairwise balancing operations can be realized by partitioning the set of edges into a number of subsets such that no two adjoining edges are in the same subset. The pairwise balancing steps along the channels in the same subset can then be performed concurrently without collisions. Such graph partition is equivalent to the problem of edge coloring of graphs [28]. Figure 2 shows examples of color graphs of a mesh and a torus. The numbers in parentheses are the assigned chromatic indices. An alternative approach to parallelizing load balancing operations is random matching which was used in [29].

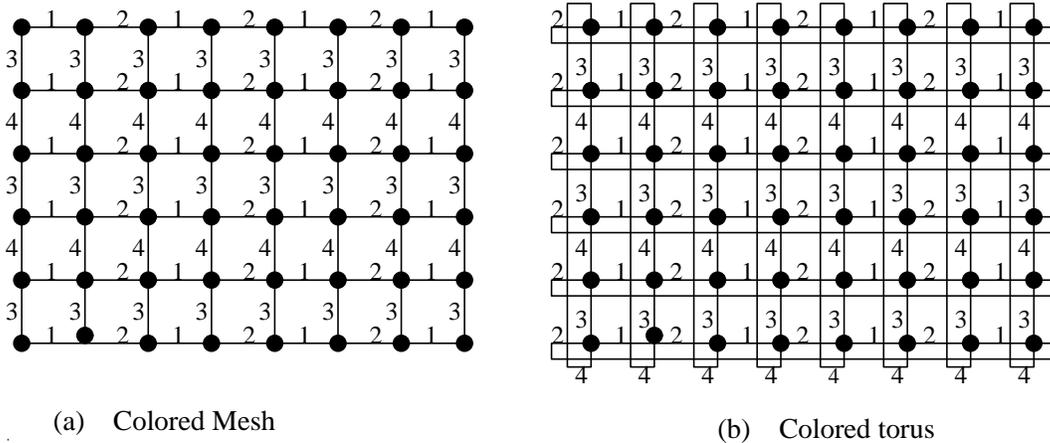


Figure 2: Examples of colored mesh and torus

3.2 The Diffusion Method

With the diffusion method, any processor which invokes a load balancing operation compares its workload with those of its nearest neighbors, and then gives away or takes in certain amount of workload with respect to each of nearest neighbors. The diffusion operator in a processor i can be written in the form

$$f_i(\cdot) \equiv w_i + \sum_{j \in \mathcal{A}(i)} \alpha_{ij}(w_j - w_i) \quad (4)$$

where $0 < \alpha_{ij} < 1$, called the diffusion parameter, is predefined to dictate the portion to be migrated between any two processors. Processor i apportions excess workload $|w_i - w_j|$ to processor j if $w_i > w_j$, or fetches some workload from processor j otherwise. Clearly, a load balancing operation with the diffusion method requires only one communication step in the all-port communication model, but $d(i)$ steps in the one-port communication model.

As in the dimension-exchange method, the efficiency of the diffusion method is determined by the diffusion parameter. Following are two common choices of the parameter.

1. *Local average diffusion* (ADF) takes an average of the workload of neighboring processors by setting $\alpha_{ij} = \frac{1}{1+d(i)}$ [12, 13, 14]; the torus is regular in that all processors have the same degree. The mesh is approximately regular when its size is large. For simplicity, we use a single value $\alpha = \frac{1}{1+d}$ to cover all communication channels in the mesh and in the torus.
2. *Optimally-tuned diffusion* (ODF) takes certain specific parameter values for maximizing efficiency in static and synchronous balancing [8]. As in the dimension-exchange method, the optimal diffusion parameter depends on the topology and the size of the underlying network. Let $k = \max\{k_1, k_2, \dots, k_n\}$ in the $k_1 \times k_2 \times \dots \times k_n$ mesh and torus. Then, their optimal choices were shown, in [8, 15], to be
 - $\alpha = 1/2n$ in the mesh,
 - $\alpha = 1/(2n + 1 - \cos(2\pi/k))$ in the torus,
 - $\alpha = 1/(n + 1)$ in the n -D hypercube.

4 Asynchronous Implementations

In an asynchronous implementation of load balancing, processors perform balancing operations discretely based on their own local workload distributions and invocation policies. Since load balancing algorithms can be treated as orthogonal to invocation policies, we consider the load balancing operations of the processors in one time step so as to isolate their effects on the workload variance from the effects of invocation policies. We focus on the static situation of load balancing in which the underlying computation in a processor is suspended while the processor is performing load balancing operations. The dynamic situation presents only a few relatively minor differences to the analysis of the effects of load balancing.

Let ν^0 be the original system workload variance when $t = 0$, and ν^1 be the system workload variance when $t = 1$. Our comparison will be made between ν_{ade}^1 , ν_{ode}^1 , ν_{adf}^1 , and ν_{odf}^1 which are the results from various load balancing operations.

Theorem 4.1 *Suppose processors are running an asynchronous load balancing process under Assumption 2.1. Then, $E[\nu_{ade}^1] \leq E[\nu_{df}^1]$ in the one-port communication model, while $E[\nu_{df}^1] \leq E[\nu_{ade}^1]$ in the all-port communication model. Moreover, $E[\nu_{adf}^1] \leq E[\nu_{odf}^1]$ in chain and ring networks, but $E[\nu_{odf}^1] \leq E[\nu_{adf}^1]$ in two- or higher- dimensional meshes and tori. In addition, $E[\nu_{ade}^1] \leq E[\nu_{ode}^1]$ in the all-port communication model.*

This theorem says that the dimension-exchange and the diffusion methods are suitable for the one-port and the all-port communication models, respectively. More specifically, it reveals

that the ODF algorithm outperforms the ADF algorithm in higher dimensional meshes and tori although the ODF was originally proposed for use in synchronous global balancing.

The theorem is proved through the derivation of the closed form of each variance $E[\nu^1]$. The calculation of $E[\nu^1]$ is based on a lemma concerning the sample variance of a combination of random variables in a sample set, which we present without proof. It can be easily shown using fundamental statistical theories.

Lemma 4.1 *Suppose that $\zeta_1, \zeta_2, \dots, \zeta_N$ are N i.i.d. random variables with variance σ^2 , and $\bar{\zeta} = \sum_{i=1}^N \zeta_i$. Then,*

1. for any k , $1 \leq k \leq N$,

$$E(|\sum_{i=1}^k a_i \zeta_i - \bar{\zeta}|^2) = (\sum_{i=1}^k a_i^2 - \frac{1}{N})\sigma^2, \quad (5)$$

where $0 < a_i < 1$ satisfies $\sum_{i=1}^k a_i = 1$; and the variance is minimized at $a_i = 1/k$ for a given k .

2. for any k_1 and k_2 and $1 \leq k_1 \leq k_2 \leq N$,

$$E(|\sum_{i=1}^{k_1} a_i \zeta_i - \bar{\zeta}|^2) \geq E(|\sum_{j=1}^{k_2} b_j \zeta_j - \bar{\zeta}|^2) \quad (6)$$

where $0 < a_i < 1$ satisfies $\sum_{i=1}^{k_1} a_i = 1$ and $0 < b_j < 1$ satisfies $\sum_{j=1}^{k_2} b_j = 1$.

Proof of Theorem 4.1 At certain time in an asynchronous load balancing process, there might be more than one processor that are invoking load balancing within their neighborhoods simultaneously. Let $\tilde{\mathcal{A}}(i) = \{i\} \cup \mathcal{A}(i)$ denote the balancing domain of an invoker processor i . The balancing domains of concurrent invokers may overlap or may be separated from each other. As a whole, those processors that are running load balancing processes are partitioned into a number of separated spheres, some of which are singular balancing domains and some are unions of overlapping domains. Processors in different spheres perform load balancing operations independently, while processors in the same sphere perform load balancing in a synchronous manner.

Suppose initially there are m independent balancing spheres in the system, denoted by $\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_m$. Then, by the definition of the workload variance ν and Assumption 2.1, we have

$$\begin{aligned} E[\nu^1] &= E(\sum_{i=1}^N |w_i^1 - \bar{w}^1|^2) \\ &= \sum_{i=1}^N E(|w_i^1 - \bar{w}^1|^2) \\ &= \sum_{j=1}^m \sum_{i \in \mathcal{B}_j} E(|w_i^1 - \bar{w}^1|^2) + \sum_{i \notin \cup_{j=1}^m \mathcal{B}_j} E(|w_i^1 - \bar{w}^1|^2) \\ &= \sum_{j=1}^m \sum_{i \in \mathcal{B}_j} E(|w_i^1 - \bar{w}^1|^2) + (N - N')(1 - \frac{1}{N})(\sigma_0^2 + \sigma^2), \end{aligned} \quad (7)$$

where $N' = |\cup_{i=1}^m \mathcal{B}_i|$ is the number of processors involved in load balancing. The last term of (7) is due to the underlying computational operations. It is a constant for a given N' and independent of the topological relationships among the N' processors. The first term of (7) is due to load balancing operations in all separated balancing spheres. It is a simple arithmetic sum of workload variance of each sphere, $\sum_{i \in \mathcal{B}_j} E(|w_i^1 - \bar{w}^1|^2)$. As a whole, Eq. (7) implies that the expected value of the system workload variance is influenced independently by load balancing operations within different balancing spheres. Therefore, it suffices to compare the effects of load balancing algorithms within different spheres using Lemma 4.1.

Case 1: Load Balancing in a Singular Balancing Domain

We first consider load balancing in spheres of singular balancing domains. Suppose \mathcal{B}_1 is such a sphere, and without loss of generality, $\mathcal{B}_1 = \tilde{\mathcal{A}}(1) = \{1, 2, 3, \dots, d+1\}$. That is, processor 1 invokes a load balancing operation within its d neighbors which are labeled from 2 to $d+1$. Let $X = \sum_{i=1}^{d+1} E(|w_i^1 - \bar{w}^1|^2)$, denoting the expected value of workload variance of sphere \mathcal{B}_1 . Then, with the diffusion algorithm, the workloads of processors at the end of a diffusion operation are given, according to Eq.(4), by

$$w_i^1 = \begin{cases} (1 - d\alpha)w_1^0 + \alpha \sum_{j=2}^{d+1} w_j^0 & \text{if } i = 1; \\ \alpha w_1^0 + (1 - \alpha)w_i^0 & \text{if } 2 \leq i \leq d+1; \end{cases} \quad (8)$$

Invoking Lemma 4.1 on each component w_i^1 , we have that

$$\begin{aligned} X_{df} &= \sum_{i=1}^{d+1} E(|w_i^1 - \bar{w}^1|^2) \\ &= E(|(1 - d\alpha)w_1^0 + \alpha \sum_{i=2}^{d+1} w_i^0 - \bar{w}^0|^2) + \sum_{i=2}^{d+1} E(|\alpha w_1^0 + (1 - \alpha)w_i^0 - \bar{w}^0|^2) \\ &= [d\alpha^2 + (1 - d\alpha)^2 - 1/N]\sigma_0^2 + d[\alpha^2 + (1 - \alpha)^2 - 1/N]\sigma_0^2 \\ &= (d^2\alpha^2 + 3d\alpha^2 - 4d\alpha + d + 1 - \frac{d+1}{N})\sigma_0^2. \end{aligned} \quad (9)$$

It can be easily verified that X_{df} , as a convex function of α , is minimized at $\alpha = 2/(3+d)$. Let $\alpha_{opt} = 2/(3+d)$. We replace α by α_{opt} in the expression of X_{df} , and obtain

$$X_{df}(\alpha_{opt}) = (\frac{d^2+3}{d+3} - \frac{d+1}{N})\sigma_0^2. \quad (10)$$

Recall that $\alpha_{adf} = 1/(d+1)$, and that $\alpha_{odf} = 1/d$ in a mesh and $\alpha_{odf} = 1/(d+1 - \cos(2\pi/k))$ in a torus, where k is the maximum dimensional order of the torus. It follows that

- in the case of a chain (*i.e.*, the 1-D mesh) where $d = 2$, $\alpha_{adf} < 2/(3+d) < \alpha_{odf}$ and $|\alpha_{adf} - \alpha_{opt}| < |\alpha_{odf} - \alpha_{opt}|$;
- in the case of a ring (*i.e.*, the 1-D torus) where $d = 2$, $\alpha_{adf} < \alpha_{opt} < \alpha_{odf}$ and $|\alpha_{adf} - \alpha_{opt}| < |\alpha_{odf} - \alpha_{opt}|$, for $k \geq 12$;
- in the case of higher dimensional meshes and tori where $d \geq 4$, $\alpha_{adf} < \alpha_{odf} < \alpha_{opt}$.

Consequently, with the diffusion method,

$$\begin{cases} X_{odf} \geq X_{adf} \geq X_{df}(\alpha_{opt}) & \text{if } d = 2 \text{ and } k \geq 12, \\ X_{adf} \geq X_{odf} \geq X_{df}(\alpha_{opt}) & \text{if } d \geq 4 \end{cases} \quad (11)$$

With the dimension-exchange method, processor 1 is assumed to perform pairwise load balancing with processors 2, 3, ..., $d + 1$ in turn in a dimension-exchange load balancing operation. Assume the underlying system is in the one-port communication model. Then, the workload generation/consumption ratio in a round of pairwise balancing steps has the same statistical characteristics as those in a diffusion operation. Consequently, the processors' workloads at the end of a dimension-exchange operation are given, according to Eq.(3), by

$$w_i^1 = \begin{cases} (1 - \lambda)^d w_1^0 + \lambda \sum_{j=0}^{d-1} (1 - \lambda)^j w_{d-j+1}^0 & \text{if } i = 1; \\ (1 - \lambda)w_2^0 + \lambda w_1^0 & \text{if } i = 2; \\ (1 - \lambda)w_i^0 + \lambda(1 - \lambda)^{i-2} w_1^0 + \lambda^2 \sum_{j=0}^{i-3} (1 - \lambda)^j w_{i-1-j}^0 & \text{if } 3 \leq i \leq d + 1; \end{cases} \quad (12)$$

Invoking Lemma 4.1 on each component w_i^1 , we have that

$$\begin{aligned} X_{de} &= \sum_{i=1}^{d+1} E(|w_i^1 - \bar{w}^1|^2) \\ &= [(1 - \lambda)^{2d} + \lambda^2 \sum_{j=0}^{d-1} (1 - \lambda)^{2j} - 1/N] \sigma_0^2 + [(1 - \lambda)^2 + \lambda^2 - 1/N] \sigma_0^2 \\ &\quad + \sum_{i=3}^{d+1} [(1 - \lambda)^2 + \lambda^2 (1 - \lambda)^{2(i-2)} + \lambda^4 \sum_{j=0}^{i-3} (1 - \lambda)^{2j} - (d - 1)/N] \sigma_0^2 \\ &= [d(1 - \lambda)^2 + 2\lambda^2 \frac{1 - (1 - \lambda)^{2d}}{1 - (1 - \lambda)^2} + \lambda^4 \frac{d - 1 - (1 - \lambda)^2 \frac{1 - (1 - \lambda)^{2d-2}}{1 - (1 - \lambda)^2}}{1 - (1 - \lambda)^2} + (1 - \lambda)^{2d} - \frac{d+1}{N}] \sigma_0^2. \end{aligned}$$

In particular, substituting $1/2$ for the λ in the expression of X_{de} , calling it X_{ade} , leads to that

$$X_{ade} = \left(\frac{3d + 5 + 2^{2-2d}}{9} - \frac{d + 1}{N} \right) \sigma_0^2. \quad (13)$$

From (10) and (13), it is known that $X_{ade} \leq X_{df}(\alpha_{opt})$. It is thus proved that in the one-port communication model,

$$X_{ade} \leq X_{df}. \quad (14)$$

In the all-port communication model, a dimension-exchange pairwise balancing step takes as much time as a diffusion load balancing operation. That is, in a time step of the diffusion method, a processor balances with only one of its neighbors with the dimension-exchange method. It results in that

$$X_{de} = 2[(1 - \lambda)^2 + \lambda^2 - \frac{1}{N}] \sigma_0^2 + (d - 1)(1 - \frac{1}{N}) \sigma_0^2.$$

Consequently, X_{ade} is less than X_{ode} but larger than X_{df} .

Case 2: Load Balancing in a Union of Overlapping Domains

We now consider load balancing in spheres which are unions of overlapping balancing domains. A balancing sphere can be a union of any number of overlapping domains. In consideration of the likelihood that few processors will be invoking load balancing simultaneously in asynchronous implementations, we focus on the union of two balancing domains only. Figure 3 illustrates three topological relationships between a pair of processors which have overlapping balancing domains in 2-D meshes and tori. The triangles are invokers of load balancing processes and the dots are processors being involved in load balancing.

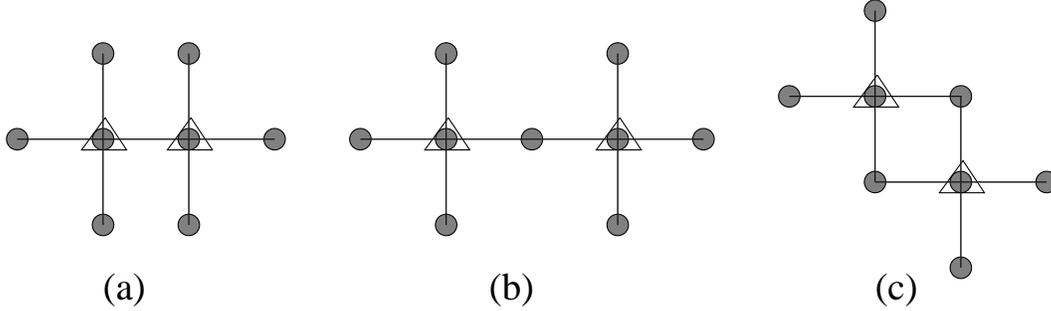


Figure 3: Illustrations of overlapping balancing domains

Assume \mathcal{B}_2 is a union of balancing domains of processors j_1 and j_2 . That is, $\mathcal{B}_2 = \tilde{\mathcal{A}}(j_1) \cup \tilde{\mathcal{A}}(j_2)$. Let Y denote the expected value of the workload variance of \mathcal{B}_2 , *i.e.*, $Y = \sum_{i \in \mathcal{B}_2} E(|w_i^1 - \bar{w}^1|^2)$. Suppose processors j_1 and j_2 have the same number of direct neighbors. Then, in the case that processors j_1 and j_2 are directly connected, as in Figure 3(a), we have that in the diffusion method,

$$Y_{df} = 2X_{df} - 2[(1 - \alpha)^2 + \alpha^2 - \frac{1}{N}] \sigma_0^2, \quad (15)$$

and in the dimension-exchange method,

$$Y_{de} \leq 2X_{de} - 2[(1 - \lambda)^2 + \lambda^2(1 - \lambda)^{2(d-1)} + \lambda^4 \frac{1 - (1 - \lambda)^{2(d-1)}}{1 - (1 - \lambda)^2} - \frac{1}{N}] \sigma_0^2. \quad (16)$$

In particular,

$$Y_{ade} \leq 2X_{ade} - 2\left(\frac{1}{3} + \frac{2}{3 \times 2^{2d}} - \frac{1}{N}\right) \sigma_0^2. \quad (17)$$

The equation of the diffusion method is due to the fact that each processor in $\tilde{\mathcal{A}}(j_1) \setminus \{j_2\}$ (or $\tilde{\mathcal{A}}(j_2) \setminus \{j_1\}$) changes its workload in the same way as in load balancing within a singular balancing domain $\tilde{\mathcal{A}}(j_1)$ (or $\tilde{\mathcal{A}}(j_2)$). The reasons of the inequality of the Y_{de} in the dimension-exchange method are as follows. With the dimension-exchange method, both processors j_1 and j_2 perform pairwise balancing operations with their neighbors in turn according to orders which are preset through edge-coloring of the system graph. The change of the workload distribution in \mathcal{B}_2 is thus influenced by the execution order across the communication channels. Suppose the channel (j_1, j_2) is indexed as c^{th} . Without loss of generality, we relabel the processor j_1 as 1, processor j_2 as $c + 1$, and other neighboring processors of processor j_1 as 2 to $d + 1$ excluding $c + 1$. Then, it is clear that processors from 2 to c change their workloads in the same way as their counterparts if they are performing load balancing within a singular domain $\tilde{\mathcal{A}}(i)$ alone, while the behaviors of other processors will be influenced by processors

in $\tilde{\mathcal{A}}(j) \setminus \{i\}$. From Lemma 4.1 (2), it is also known that each processor i , $i > c$, will possess less workload variance $E(|w_i^1 - \bar{w}^1|^2)$ in a union of overlapping domains than in $\tilde{\mathcal{A}}(i)$ alone. Since $E(|w_{d+1}^1 - \bar{w}^1|^2) \leq E(|w_i^1 - \bar{w}^1|^2)$ for $i > c$, the bound of Y_{de} is hence obtained.

It can be easily verified that Y_{df} is minimized at $\alpha = (2d - 1)/(d^2 + 3d - 2)$. Let α_{opt} be the optimal choice of α . Then,

$$Y_{df}(\alpha_{opt}) = d - \frac{4d^2 - 4d + 1}{d^2 + 3d - 2}. \quad (18)$$

As in the case of singular balancing domain, it can be shown that

- in the case of a chain (*i.e.*, 1-D mesh) where $d = 2$, $\alpha_{adf} < \alpha_{opt} < \alpha_{odf}$ and $|\alpha_{adf} - \alpha_{opt}| < |\alpha_{odf} - \alpha_{opt}|$;
- in the case of a ring (*i.e.*, 1-D torus) where $d = 2$, $\alpha_{adf} < \alpha_{opt} < \alpha_{odf}$ and $|\alpha_{adf} - \alpha_{opt}| < |\alpha_{odf} - \alpha_{opt}|$, for $k \geq 6$;
- in the case of higher dimensional meshes and tori where $d \geq 4$, $\alpha_{adf} < \alpha_{odf} < \alpha_{opt}$.

Consequently, with the diffusion method,

$$\begin{cases} Y_{odf} \geq Y_{adf} \geq Y_{df}(\alpha_{opt}) & \text{if } d = 2 \text{ and } k \geq 6, \\ Y_{adf} \geq Y_{odf} \geq Y_{df}(\alpha_{opt}) & \text{if } d \geq 4 \end{cases} \quad (19)$$

On the other hand, the comparison between Y_{ade} of Eq.(17) and $Y_{df}|_{\alpha=\alpha_{opt}}$ of Eq.(18) yields $Y_{ade} \leq Y_{df}(\alpha_{opt})$.

In cases that processors i and j are nonadjacent, as illustrated in Figure 3(b) and 3(c), there are at most two processors in the intersect of their balancing domains in the mesh and torus networks. Let s be the cardinality of the intersect $\tilde{\mathcal{A}}(i) \cap \tilde{\mathcal{A}}(j)$, $s = 1$ or 2 . Then, with the diffusion method,

$$\begin{aligned} Y_{df} &= 2X_{df} - 2s[(1 - \alpha)^2 + \alpha^2 - \frac{1}{N}]\sigma_0^2 + s[(1 - 2\alpha)^2 + 2\alpha^2 - \frac{1}{N}]\sigma_0^2 \\ &= 2X_{df} - s[(1 - 2\alpha^2) - \frac{1}{N}]\sigma_0^2; \end{aligned} \quad (20)$$

and with the ADE algorithm,

$$Y_{ade} \leq 2X_{ade} - s[\frac{1}{3} + \frac{2}{3 \times 2^{2d}} - \frac{1}{N}]\sigma_0^2. \quad (21)$$

Similarly to the case of singular balancing domain, we have the result that $Y_{ade} \leq Y_{df}$, $Y_{odf} \leq Y_{adf}$ in case $d = 2$, and $Y_{adf} \leq Y_{odf}$ in case $d \geq 4$.

Notice that the preceding analysis of the dimension-exchange method is implicitly based on the assumption of one-port communication model. In the all-port communication model, a dimension-exchange pairwise balancing step corresponds to a diffusion balancing operation. Because two pairwise balancing operations in a union of two balancing domains can be performed concurrently, we thus have

$$Y_{de} = 4[(1 - \lambda)^2 + \lambda^2 - \frac{1}{N}]\sigma_0^2 + (2d - 4 - s)(1 - \frac{1}{N})\sigma_0^2,$$

where $s = 1$ or 2 . Obviously, Y_{ade} is less than Y_{ode} but larger than Y_{df} .

The theorem is then proved. ■

Note that even though the proof of the theorem assumes static workload behaviors, the theorem still holds in the dynamic situation. Consider processors in balancing sphere \mathcal{B} . Since the workloads generated/consumed from time 0 to time 1 in any processor i , $i \in \mathcal{B}$, will not be considered in its load balancing operation at time step 1, the operation in the dynamic situation then results in an workload variance $E(|w_i^1 - \bar{w}^1|^2) + \sigma^2$, where $E(|w_i^1 - \bar{w}^1|^2)$ is the processor's workload variance in the static situation. As a whole, the accumulative workload variance of processors in balancing sphere \mathcal{B} in the dynamic situation is $\sum_{i \in \mathcal{B}} E(|w_i^1 - \bar{w}^1|^2) + N'\sigma^2$, where $N' = |\mathcal{B}|$. The added term is a constant for a given N' and independent of the load balancing algorithm used. Hence, the arguments in the proof of the theorem are valid in the dynamic situation.

To conclude this section, we remark that a diffusion load balancing operation averages the workload of a processor, say processor 1, and its surrounding d processors, labeled from 2 to $d + 1$, in a simple way that processor 1 gives $\alpha(w_1 - w_i)$ loads to processor i , in the case of $w_1 > w_i$, and takes $\alpha(w_i - w_1)$ load from processor i , otherwise ($2 \leq i \leq d + 1$). In a singular balancing domain, there might be a variant of the ADF algorithm which strives for local load balancing. Specifically, processor 1 calculates the local average \bar{w}_1 as

$$\bar{w}_1 = \frac{w_1 + \sum_{2 \leq i \leq d+1} w_i}{1 + d},$$

and then gives or takes $|w_i - \bar{w}_1|$ loads to or from processor i according to whether processor i is deficient or not. After such an operation, each processor i , $2 \leq i \leq d + 1$ ends up with the same workload as processor 1. Consequently, the expected workload variance of the domain $\sum_{i=1}^{d+1} E(|w_i^1 - \bar{w}^1|^2)$ becomes

$$\left(1 - \frac{d+1}{N}\right)\sigma_0^2,$$

which is obviously smaller than that of the ADE method even in the one-port communication model. Although it incurs more overheads than an ODF or ADF operation, such a variant of diffusion operation is preferred in singular balancing domains. However, it may not be effective in balancing spheres where a number of balancing domains overlap with each other because processors in such a sphere are unable to balance their workloads with all the processors in such an operation.

5 Synchronous Implementations

In a synchronous implementation of load balancing, all processors perform load balancing operations concurrently and continuously for a time period in order to achieve a global balanced state in the static situation or to keep the varying system workload variance bounded in the dynamic situation.

The synchronous implementation of the diffusion and the dimension-exchange methods can be modeled by linear iterative processes, as illustrated in [12, 8, 5]. From Eq. (4), the change of the workload distribution at time t in the diffusion method can be modeled by the equation

$$W^{t+1} = DW^t + \Phi^t, \quad (22)$$

where D , called a diffusion matrix, is given by

$$D_{ij} = \begin{cases} \alpha & \text{if processors } i \text{ and } j \text{ are directly connected;} \\ 1 - d(i)\alpha & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

With the above formulation, the features of the synchronous implementation of the diffusion method are fully captured by the iterative process governed by D .

Let κ be the minimum number of colors required for edge coloring of the system graph. Then, from Eq. (3), the change of the workload distribution at time t in the dimension-exchange method can be modeled by the equation

$$W^{t+1} = MW^t + \Phi^t \quad (23)$$

where M , called the dimension-exchange matrix, is given by

$$M = M_\kappa \times M_{\kappa-1} \times \dots \times M_1.$$

Each M_c ($1 \leq c \leq \kappa$) reflects the change of the workload distribution of the system at pairwise balancing step c of time t . Thus, the features of the synchronous implementation of the dimension-exchange method are fully captured by the iterative process governed by M .

5.1 Static Situation

In a static synchronous load balancing process, all processors are assumed to perform load balancing operations simultaneously and all computational operations are suspended. This situation is true of periodic load balancing, as experimented in [30, 31, 25, 19]. The efficiency of a load balancing algorithm in this situation is reflected by the number of communication steps required for arriving at a global balanced state from any initial load distribution.

Let F be either the dimension-exchange matrix M or the diffusion matrix D . Then, $W^t = F^t W^0$, where $F^t = \underbrace{F \times F \times \dots \times F}_{t \text{ times}}$. Since $\overline{W}^t = \overline{W}^0$ in the static situation, and $F^t \overline{W}^t = \overline{W}^t$, it follows that

$$W^t - \overline{W}^t = F(W^{t-1} - \overline{W}^{t-1}) = F^t(W^0 - \overline{W}^0).$$

Then, by the definition of the workload variance, we have

$$\nu^t = \|W^t - \overline{W}^t\|^2 = \|F^t(W^0 - \overline{W}^0)\|^2 \leq \gamma^{2t}(F)\nu^0,$$

where $\gamma(F)$ is the sub-dominant eigenvalue of F in modulus. It says that the workload variance is reduced geometrically, and its scale factor is upper bounded by $\gamma(F)$. The bound is tight, and ν^t satisfies¹

$$\nu^t \simeq \gamma^{2t}(F)\nu^0 \quad \text{for large } t. \quad (24)$$

The sub-dominant eigenvalue in modulus $\gamma(F)$ is thus referred also as the convergence factor of a load balancing algorithm.

¹By $g(t) \simeq h(t)$ for large t , we mean that $g(t)/h(t) \rightarrow 1$ for large t .

Let T be the number of iteration steps required to reduce the workload variance of an initial state to some prescribed bound τ . Then, from Eq. 24, it follows that

$$T = \frac{\ln \tau - \ln \nu^0}{2 \ln \gamma(F)}. \quad (25)$$

Hence,

$$T = O(1/\ln \gamma(F)). \quad (26)$$

The convergence factors of the dimension-exchange and the diffusion methods were evaluated by a number of researchers. In [12], Boillat presented the convergence factors of the ADF algorithm, γ_{adf} , when applied to a broad variety of structures. In [5, 15], Xu and Lau analyzed the effects of the dimension-exchange and the diffusion parameters on the efficiency of load balancing, and proposed the ODE and ODF algorithms by choosing the optimal values for the parameters α and λ . The corresponding convergence factors, γ_{ode} and γ_{odf} , are hence readily available from their proofs. Also, the convergence factor γ_{ade} can be derived easily from the work. We summarize the convergence factors in Table 1.

	DE method		Diffusion method	
	ADE	ODE	ADF	ODF
torus	$\cos^2(2\pi/k)$	$\frac{1-\sin(2\pi/k)}{1+\sin(2\pi/k)}$	$\frac{2n-1+2\cos(2\pi/k)}{2n+1}$	$\frac{2n-1+\cos(2\pi/k)}{2n+1-\cos(2\pi/k)}$
mesh	$\cos^2(\pi/k)$	$\frac{1-\sin(\pi/k)}{1+\sin(\pi/k)}$	$\frac{2n-1+2\cos(\pi/k)}{2n+1}$	$\frac{n-1+\cos(\pi/k)}{n}$

Table 1: Convergence factors of the dimension-exchange and the diffusion methods, where k is the maximum number of nodes over all dimensions of an n -D network.

Notice that the convergence factor is in iteration steps, each of which is what we called a load balancing operation before. In the one-port communication model, such an operation in both the dimension-exchange and the diffusion methods requires $2n$ communication steps in an n -D network. In the all-port communication model, a diffusion load balancing operation requires only one communication step while a dimension-exchange operation still requires $2n$ steps. Therefore, Table 1 and the Eq. (26) lead to Table 2. It presents the time complexities in communication steps necessary for various load balancing algorithms in both one-port and all-port communication models.

	ADE		ODE		ADF		ODF	
	1-port	*-port	1-port	*-port	1-port	*-port	1-port	*-port
torus	$O(nk^2)$	$O(nk^2)$	$O(nk)$	$O(nk)$	$O(n^2k^2)$	$O(nk^2)$	$O(n^2k^2)$	$O(nk^2)$
mesh	$O(nk^2)$	$O(nk^2)$	$O(nk)$	$O(nk)$	$O(n^2k^2)$	$O(nk^2)$	$O(n^2k^2)$	$O(nk^2)$

Table 2: Time complexities of the dimension-exchange and the diffusion methods, where k is the maximum number of nodes over all dimensions in an n -D network and $*-port$ means the all-port communication model.

The time complexities given in Table 2 are inferred from the convergence factors. For example, the $O(nk)$ estimate for the ODE algorithm follows from the following derivation.

$$\ln(\gamma_{ode}) = \ln\left(\frac{1 - \sin(2\pi/k)}{1 + \sin(2\pi/k)}\right)$$

$$\begin{aligned}
&= \ln\left(1 - \frac{2 \sin(2\pi/k)}{1 + \sin(2\pi/k)}\right) \\
&\simeq \ln\left(1 - \frac{4\pi}{k + 2\pi}\right) \quad \text{for large } k \\
&\simeq \frac{4\pi}{k + 2\pi}
\end{aligned}$$

From Eq. (26), we have $T_{ode} = O(k)$ in balancing operations. Since an ODE load balancing operation requires $O(n)$ communication steps in both communication models, the estimate $O(nk)$ is thus proved.

The entries of the table show the following.

Theorem 5.1 *Suppose processors are running synchronous load balancing processes in the static situation. Then, both the ADE and the ODE algorithms converge asymptotically faster than the diffusion method in the one-port communication model. In the all-port communication model, the ODE algorithm converges also faster than the other three algorithms by a factor of k .*

5.2 Dynamic Situation

In dynamic synchronous implementations, all processors are performing load balancing and computation concurrently. Dynamic load balancing in this situation aims at keeping the variance of processors' workloads bounded as tightly as possible. The performance of the synchronous implementation of the diffusion method in the dynamic situation has been evaluated in [8, 13, 14]. In [8], Cybenko showed that the diffusion method keeps the asymptotic variance bounded. He also proved that the asymptotic variance from the diffusion method is larger than the variance from the ADE algorithm when both are applied to the hypercube network. Given this, we are still unable to draw a conclusion regarding the superiority of the ADE algorithm in terms of the balance quality during load balancing. In [13], Hong, Tan and Chen reported a constant bound for the workload variance when the ADF algorithm runs in the hypercube network. This result was extended later by Qian and Yang to generalized hypercubes and mesh structures [14]. Although the bounds they derived are independent of time, they are too loose to be used for the comparison of balance qualities during load balancing. Also, the approaches used in [13, 14] are unsuitable for the analysis of the dimension-exchange method because of their different operational behaviors.

In this subsection, we develop a new approach for analyzing the balance qualities of different algorithms. We present a closed form of the workload variance when a load balancing process runs in the torus and the hypercube networks. The approach is not applicable to the case of the mesh networks as they are not regular networks. Nevertheless, since an n -D mesh has only a fraction of its processors whose degree is smaller than $2n$, our results as a reasonable approximation can be applied to the mesh structure as well; this is supported by our simulation results to be presented in the next section.

Throughout the subsection, we assume load balancing in an n -D torus network, and let $d = 2n$ be the degree of the network.

Lemma 5.1 *Suppose processors are running a synchronous diffusion load balancing process under Assumption 2.1. Then, $E(W^t)$ is a uniform distribution at any time t and*

$$E[\nu_{df}^t] = (a^{t+1}\sigma_0^2 + \frac{1 - a^{t+1}}{1 - a}\sigma^2)N - (t + 1)\sigma^2 - \sigma_0^2, \quad (27)$$

where $a = (1 - d\alpha)^2 + d\alpha^2$.

Proof. The uniform distribution of $E(W^t)$ resulting from the diffusion method can be easily shown. We omit its proof because it is also available as a special case in the proof of the uniform distribution of $E(W^t)$ resulting from the dimension-exchange method in the next lemma.

Consider the expected workload variance $E[\nu_{df}^t]$. By its definition and Assumption 2.1, we have

$$\begin{aligned}
E[\nu_{df}^t] &= E(\|W^t - \overline{W}^t\|^2) \\
&= E(\|DW^{t-1} - \overline{W}^{t-1}\|^2) + E(\|\Phi^t - \overline{\Phi}^t\|^2) \\
&= E(\|D^{t+1}W^0 - \overline{W}^0\|^2) + \sum_{i=0}^t E(\|D^i\Phi^{t+1-i} - \overline{\Phi}^{t+1-i}\|^2) \\
&= N(a^{t+1} - \frac{1}{N})\sigma_0^2 + \sum_{i=0}^t (a^i - \frac{1}{N})\sigma^2 \\
&= (a^{t+1}\sigma_0^2 + \frac{1-a^{t+1}}{1-a}\sigma^2)N - (t+1)\sigma^2 - \sigma_0^2,
\end{aligned}$$

where the fourth step is based on the following observations. An operation D on the workload distribution Φ changes each of its components to become a linear combination of the component's $d+1$ sub-components with coefficients $1-d\alpha, \alpha, \alpha, \dots, \alpha$; and a sequence of operation D^t changes each component of Φ to become a linear combination of its $(d+1)^t$ sub-components. From Lemma 4.1, it is known that the variance of a combination of random variables is determined only by their combinatorial coefficients. Therefore, we have $E(\|D\Phi - \overline{\Phi}^{t+1-i}\|^2) = N(a - 1/N)\sigma^2$, and $E(\|D^t\Phi - \overline{\Phi}\|^2) = N(a^t - 1/N)\sigma^2$, where $a = (1-d\alpha)^2 + d\alpha^2$. ■

Consider the term $a = (1-d\alpha)^2 + d\alpha^2$ in Lemma 5.1. It is minimized at $\alpha = 1/(d+1)$ over all possible choices of the parameter α , which happens to be the choice of the ADF algorithm in n -D meshes and tori. Immediately, we obtain

$$E[\nu_{adf}^t] \leq E[\nu_{odf}^t]. \quad (28)$$

Next, we consider synchronous implementations of the dimension-exchange method. We present a companion to Lemma 5.1 in the following.

Lemma 5.2 *Suppose processors are running a synchronous dimension-exchange load balancing process under Assumption 2.1, except that processors generate/consume ϕ_i workload at a pairwise balancing step. Then, $E(W^t)$ is a uniform distribution at any time t , and*

$$E[\nu_{de}^t] = (sb^{td+d}\sigma_0^2 + s\frac{1-b^{td+d}}{1-b^d}\sigma^2)N - (t+1)d\sigma^2 - \sigma_0^2, \quad (29)$$

where $b = (1-\lambda)^2 + \lambda^2$ and $s = 1 + b + b^2 + \dots + b^{d-1}$.

Proof. Recall that t is the index of load balancing operations in the dimension-exchange algorithm. A load balancing operation comprises d pairwise balancing steps in both the torus and the mesh structures. To examine closely the dynamic behavior of the dimension-exchange algorithm in the level of pairwise operations, we introduce one more variable t' to denote the

index of pairwise steps. $t = 0$ if and only if $t' = 0$, and t indexes the time instances t' that are integer multiplies of d . Then, at time t' that $t' = dt$,

$$\begin{aligned}
W^{t'} &= M_d W^{t'-1} + \Phi^{t'} \\
&= M_d M_{d-1} W^{t'-2} + M_d \Phi^{t'-1} + \Phi^{t'} \\
&= \Pi_{c=d}^1 M_c W^{t'-d} + \Pi_{c=d}^2 M_c \Phi^{t'-d+1} + \dots + M_d \Phi^{t'-1} + \Phi^{t'} \\
&= M W^{t'-d} + \sum_{c=1}^d (\Pi_{j=d}^{c+1} M_j \Phi^{t'-d+c}), \tag{30}
\end{aligned}$$

where $\Pi_{j=d}^c M_j = M_d \times M_{d-1} \times \dots \times M_c$, and $\Pi_{j=d}^{d+1} M_j = 1$.

Let $\Psi^t = \sum_{c=1}^d (\Pi_{j=d}^{c+1} M_j \Phi^{t'-d+c})$ be the distribution of workloads which are generated/consumed from time $t' - d$ to t' , *i.e.*, the t^{th} dimension-exchange balancing operation. Using index t instead of t' , Eq.(30) leads to

$$\begin{aligned}
W^t &= M W^{t-1} + \Psi^t \\
&= M^t W^0 + \sum_{j=1}^t M^{t-j} \Psi^j.
\end{aligned}$$

Using the linearity of the expectation operations, E , we obtain that

$$\begin{aligned}
E(W^t) &= E(M^t W^0 + \sum_{j=1}^t M^{t-j} \Psi^j) \\
&= M^t E(W^0) + \sum_{j=1}^t M^{t-j} E(\Psi^j) \\
&= \mu_0 \mathbf{u} + dt \mu \mathbf{u},
\end{aligned}$$

where \mathbf{u} is a unitary vector of size N . It is a uniform distribution. The first part of the lemma is thus proved.

Next, we consider the workload variance at time t , $E[\nu_{de}^t]$. Let $\bar{\Psi}^t$ be the uniform distribution of workloads that are generated/consumed in the round t . Then, $\bar{\Psi}^t = \sum_{c=1}^d \bar{\Phi}^{t'-d+c}$, and $\bar{W}^t = \bar{W}^{t-1} + \bar{\Psi}^t$. By the definition of workload variance, we have

$$\begin{aligned}
E[\nu_{de}^t] &= E(\|W^t - \bar{W}^t\|^2) \\
&= E(\|M W^{t-1} - \bar{W}^{t-1}\|^2) + E(\|\Psi^t - \bar{\Psi}^t\|^2) \\
&= E(\|M^{t+1} W^0 - \bar{W}^0\|^2) + \sum_{i=0}^t E(\|M^i \Psi^{t+1-i} - \bar{\Psi}^{t+1-i}\|^2).
\end{aligned}$$

To prove the lemma regarding $E[\nu_{de}^t]$, it suffices to show that for $0 \leq i \leq t$,

$$E(\|M^i \Psi^{t+1-i} - \bar{\Psi}^{t+1-i}\|^2) = b^{id} s N \sigma^2 - d \sigma^2.$$

It can be shown by induction. We first consider $E(\|\Psi - \bar{\Psi}\|^2)$. It is the workload variance augmented in a round of dimension-exchange pairwise balancing operations. By the definition

of Φ , we have that

$$\begin{aligned}
E(\|\Psi^t - \bar{\Psi}^t\|^2) &= E\left(\left\|\sum_{c=1}^d (\Pi_{j=d}^{c+1} M_j \Phi^{t'-d+c} - \bar{\Phi}^{t'-d+c})\right\|^2\right) \\
&= \sum_{c=1}^d [E(\|\Pi_{j=d}^{c+1} M_j \Phi^{t'-d+c} - \bar{\Phi}^{t'-d+c}\|^2)] \\
&= \sum_{c=1}^{d-1} (b^c N - 1)\sigma^2 + (N - 1)\sigma^2 \\
&= sN\sigma^2 - d\sigma^2,
\end{aligned}$$

where the second step is due to the fact that $\Pi_{j=d}^{c+1} M_j \Phi^{t'-d+c} - \bar{\Phi}^{t'-d+c}$ are zero mean independent random variables for $1 \leq c \leq d$, and the third step is due to the following reasons. Each component of $\Pi_{j=d}^c M_j \Phi$ for any c , $1 \leq c < d$, is recursively a combination of two components of $\Pi_{j=d}^{c+1} M_j \Phi$ with coefficients $1 - \lambda$ and λ . It can thus be inferred that a component of $\Pi_{j=d}^c M_j \Phi$ is a combination of 2^{d-c+1} components of Φ with coefficients as follows.

	Combinatorial coefficients a_i , where $\bar{\lambda} = 1 - \lambda$								$\sum a_i^2$
$\Pi_d^d \Phi$				$\bar{\lambda}$	λ				b
$\Pi_d^{d-1} \Phi$			$\bar{\lambda}^2$	$\lambda \bar{\lambda}$	$\lambda \bar{\lambda}$	λ^2			b^2
$\Pi_d^{d-2} \Phi$	$\bar{\lambda}^3$	$\lambda \bar{\lambda}^2$	$\lambda \bar{\lambda}^2$	$\lambda^2 \bar{\lambda}$	$\bar{\lambda}^2 \lambda$	$\lambda^2 \bar{\lambda}$	$\lambda^2 \bar{\lambda}$	λ^3	b^3
...			
$\Pi_d^1 \Phi$	$\bar{\lambda}^d$	$\lambda \bar{\lambda}^{d-1}$	$\lambda^2 \bar{\lambda}^{d-2}$...		$\lambda^{d-2} \bar{\lambda}^2$	$\lambda^{d-1} \bar{\lambda}$	λ^{d-1}	b^d

Consequently, from Lemma 4.1, it follows that $E(\|\Pi_d^c M_j \Phi - \bar{\Phi}\|^2) = Nb^{d-c+1}\sigma^2 - \sigma^2$.

We proceed by induction on i . Assume $E(\|M^i \Psi^{t+1-i} - \bar{\Psi}^{t+1-i}\|^2) = b^{id} sN\sigma^2 - d\sigma^2$. We then consider $E(\|M^{i+1} \Psi^{t-i} - \bar{\Psi}^{t-i}\|^2)$. Since ϕ_i^t is assumed to be independent of time t , Φ^t is independent of t as well. Then, $E(\|M^{i+1} \Psi^{t-i} - \bar{\Psi}^{t-i}\|^2) = E(\|(\Pi_{j=d}^1 M_j) M^i \Psi^{t-i+1} - \bar{\Psi}^{t-i+1}\|^2)$. From the argument in the preceding paragraph, it is known that a sequence of suffix operators $\Pi_{j=d}^1 M_j$ redistributes the workloads of $M^i \Psi$ in such a way that each of its components becomes a combination of its 2^d components with coefficients as in the last row of the table. Consequently $E(\|M^{i+1} \Psi^{t-i} - \bar{\Psi}^{t-i}\|^2) = b^d b^{id} sN\sigma^2 - d\sigma^2 = b^{id+d} sN\sigma^2 - d\sigma^2$, which concludes the induction and proves the second part of the lemma. ■

From the lemma, it is evident that $E[\nu_{de}^t]$ is minimized at $\lambda = 1/2$ over all possible choices of the dimension-exchange parameter. Thus, we have

$$E[\nu_{ade}^t] \leq E[\nu_{ode}^t]. \quad (31)$$

We further compare $E[\nu_{ade}^t]$ with $E[\nu_{adf}^t]$ in both one-port and all-port communication models. Notice that Lemma 5.2 holds under the assumption that the workload generation/consumption ratios ϕ_i^t in each pairwise balancing step of a round of dimension-exchange operation has the same statistical characteristics as those in a diffusion operation. It is therefore fair to compare $E[\nu_{df}^t]$ of Eq. (27) with $E[\nu_{de}^t]$ of Eq. (29). Consider the all-port communication model. Substituting $1/d+1$ for α in $E[\nu_{df}^t]$ and $1/2$ for λ in $E[\nu_{de}^t]$, we obtain

$$\begin{aligned}
E[\nu_{adf}^t] &= \frac{d+1}{d} [1 - (\frac{d}{d+1})^{t+1}] N\sigma^2 - (t+1)\sigma^2 + \frac{1}{(d+1)^{t+1}} \sigma_0^2 - \sigma^2 \\
E[\nu_{ade}^t] &= (2 - \frac{1}{2^{d-1}}) \frac{1 - 1/2^{t+1}}{1 - 1/2^d} N\sigma^2 - (t+1)d\sigma^2 + (2 - \frac{1}{2^{d-1}}) \frac{1}{2^{td+d}} \sigma_0^2 - \sigma^2.
\end{aligned}$$

It can be easily verified that the coefficient of σ_0^2 in $E[\nu_{ade}^t]$ is smaller than that in $E[\nu_{adf}^t]$, and that the coefficient of σ^2 in $E[\nu_{ade}^t]$ is smaller than that in $E[\nu_{adf}^t]$ when $t \leq N/d$. Hence, for $t \leq N/d$,

$$E[\nu_{ade}^t] \leq E[\nu_{adf}^t].$$

Since $N \gg d$ in the mesh and the torus, the above relationship holds for any time instant of interest in practice.

In the case of the one-port communication model, the workload generated/consumed by a processor in a single diffusion operation is expected to be $d\mu$ with variance $d\sigma^2$. Then, $E[\nu_{adf}^t]$ of Eq. (27) becomes

$$(a^{t+1}\sigma_0^2 + \frac{1-a^{t+1}}{1-a}d\sigma^2)N - (t+1)d\sigma^2 - \sigma_0^2.$$

Clearly, $E[\nu_{ade}^t] \leq E[\nu_{adf}^t]$ at any time t . Conclusively, we obtain the following theorem.

Theorem 5.2 *Suppose processors are running synchronous dimension-exchange and diffusion load balancing processes under Assumption 2.1. Then, $E[\nu_{ade}^t] \leq E[\nu_{ode}^t]$, $E[\nu_{adf}^t] \leq E[\nu_{odf}^t]$, and $E[\nu_{ade}^t] \leq E[\nu_{adf}^t]$ in both one-port and all-port communication models.*

6 Numerical Experiments

In the preceding two sections, we explored a number of relationships between the dimension-exchange and the diffusion methods with respect to their efficiencies and balancing qualities. In order to obtain an idea of the magnitude of their differences, we conducted a statistical simulation of these load balancing algorithms on various topologies and sizes of communication networks and using synthetic workload distributions. The experimental results also serve to verify the theoretical results.

The experiment includes three parts. They are a simulation of synchronous load balancing in a static workload situation, a simulation of asynchronous load balancing in the dynamic situation, and a simulation of synchronous load balancing in the dynamic situation. In each simulation, the initial workload distribution W is assumed to be a random vector, each element w of which is drawn independently from an identical uniform distribution in $[0, 1000]$. Each data point obtained in the experiment is the average of 20 runs, using different random initial workload distributions and different workload generation ratios. We also assume that the underlying system implements the all-port communication model so that a dimension-exchange balancing operation takes $2n$ diffusion operations in an n -D mesh or torus. A diffusion operation is taken as a basic time step in a load balancing process.

In the simulation of static synchronous load balancing processes, we measure the number of communication steps, denoted by T , necessary for arriving at a globally balanced state. In the simulation, we define the global balanced state to be the state in which the system workload variance is less than or equal to one. Figure 4 and Figure 5 plot the simulation results from different load balancing algorithms executed in the ring of sizes (N) varying from 2 to 128 nodes and in the 2-D mesh of sizes varying from 2×2 to 32×32 , respectively. These two figures clearly indicate that the dimension-exchange method outperforms the diffusion method even in the all-port communication model. In particular, we see that the ODE algorithm does accelerate the dimension-exchange load balancing process significantly. In the ring of 64 nodes, for example, $T_{ode} = 98$ with the ODE algorithm while $T_{ade} \simeq T_{odf} = 1305$

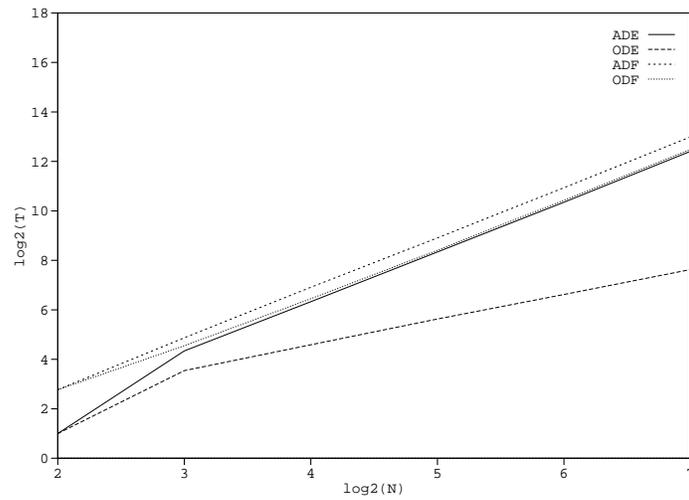


Figure 4: The number of communication steps necessary for reaching a globally balanced state during a static synchronous load balancing process in the ring of sizes varying from 4 to 128 nodes.

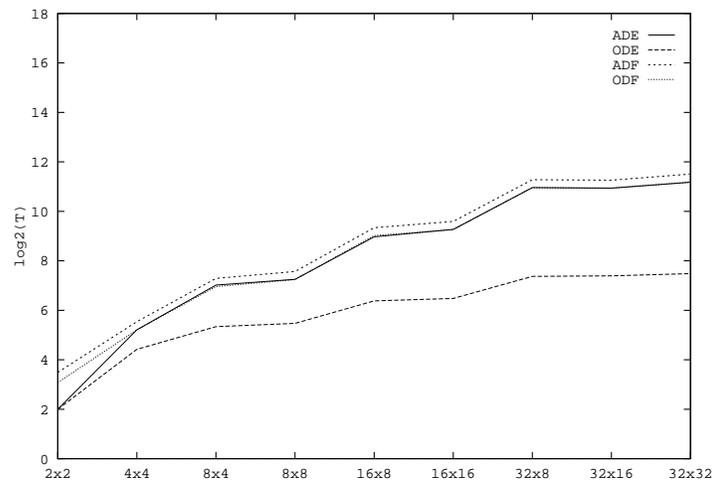


Figure 5: The number of communication steps necessary for reaching a globally balanced state during a static synchronous load balancing process in the 2-D mesh of sizes varying from 2×2 to 32×32 .

and $T_{adf} = 1684$ with the others. Its improvement over the ADE algorithm reaches as high as 92.5%. In Figure 5, we also see that the number of communication steps T in a 2-D mesh is dependent only on the size of its larger dimension and is insensitive to the size of its smaller dimension. This observation was proved to be true in both the mesh and the torus in [9]. Thus, an ODE load balancing process in a 64-ary 2-cube only requires about 196 communication steps for arriving at a global balanced state. It really puts forth the ODE algorithm as a practical method for dynamic global balancing in real multicomputers.

Furthermore, in order to examine the effects of a single load balancing operation, we plot in Figure 6 the system workload variance in the first 100 steps of various load balancing processes in the ring of 32 nodes. The figure illustrates that the ODE algorithm pulls down

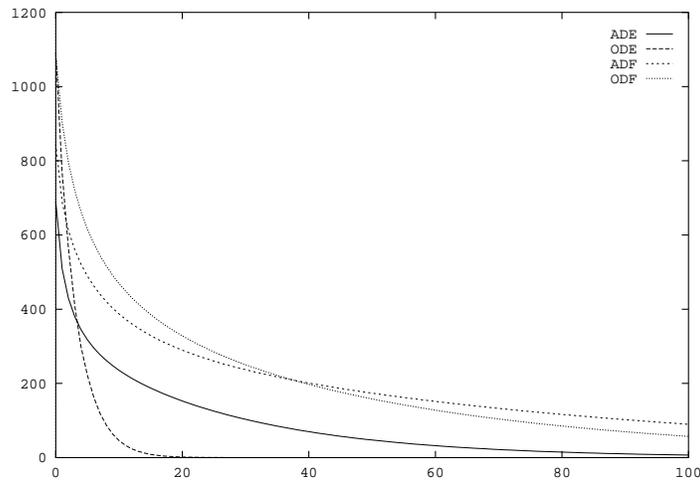


Figure 6: Reduction of the workload variance during a static synchronous load balancing process in the ring of 32 nodes.

the system workload variance sharply although its initial reduction ratio seems to be not as satisfactory as that of the ADE algorithm. The ODF and the ADF algorithms have the same relationship in their reduction ratios. This says that both the ODE and the ODF algorithms may not outperform their local average balancing counterparts in the short term.

In the dynamic situation, we assume that the expected workload generation ratio of a processor at each time step is 100 with the variance of 30 and the consumption ratio is a constant 100. In the simulation of asynchronous load balancing, we use a simple invocation policy such that once a processor's workload drops or rises beyond a pair of preset bounds, $w_{underload}$ and $w_{overload}$, the processor would activate a load balancing operation. Evidently, the pair of thresholds determine the degree of asynchronism of a load balancing process. Suppose $w_{underload}$ and $w_{overload}$ are symmetric with respect to the expected workload of a processor $E(w)$. We then measure the range between $w_{underload}$ and $w_{overload}$ by an index *range*. Since $E(w) = 500$ at any time, it follows that $w_{underload} = 500 - range/2$ and $w_{overload} = 500 + range/2$. Figures 7 and 8 plot the system workload variances resulting from different load balancing algorithms in a ring of 64 nodes and a mesh of size 16×16 for the case of $range = 600$.

From these two figures, it can be seen that the ADE algorithm reduces the initial system workload variance more rapidly than the diffusion method and keeps it bounded at a much

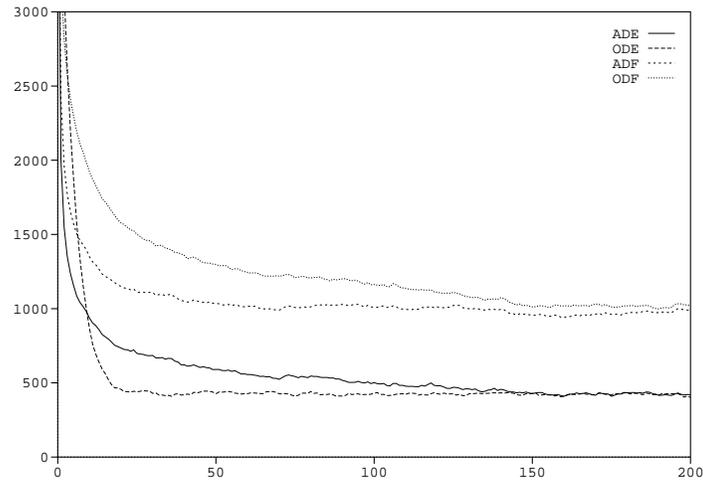


Figure 7: Change of the workload variance in the first 200 steps of a dynamic asynchronous load balancing process in the ring of size 64.

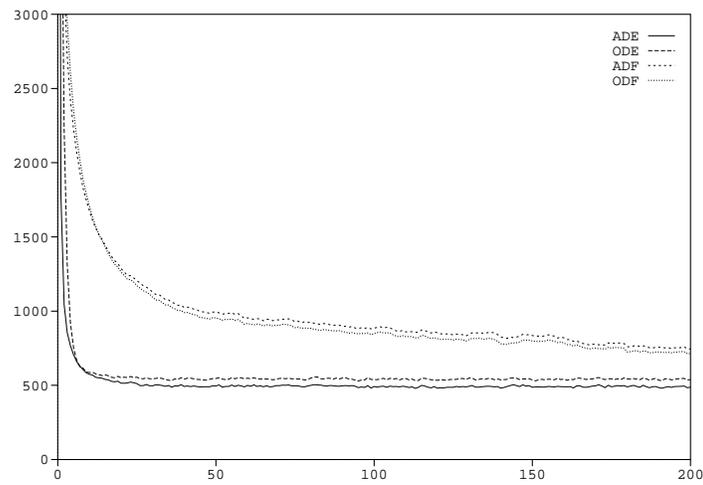


Figure 8: Change of the workload variance in the first 200 steps of a dynamic asynchronous load balancing process in the mesh of size 16×16 .

lower level. It can also be observed that both the ODE and the ODF algorithms, the optimally tuned algorithms for global synchronous load balancing, do not gain significant benefits in asynchronous implementations.

Synchronous implementations of load balancing are special cases of asynchronous implementations in which *range* is set to zero so that all processors participate in load balancing simultaneously. The simulation of synchronous implementations in the static situation was conducted in the first experiment, and its results in a ring of 32 nodes are reported in Figure 6. Figures 9 and 10 present the simulation results of dynamic synchronous implementations in the 16×16 torus and the 16×16 mesh. In agreement with the findings from Figure 6, Fig-

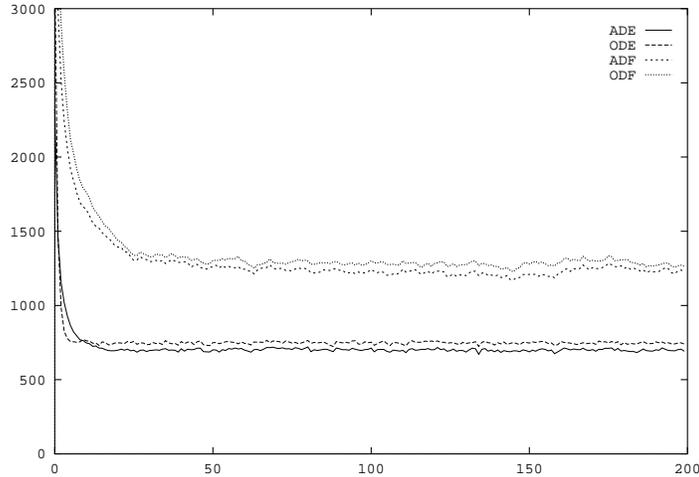


Figure 9: Change of the workload variances during a dynamic synchronous load balancing process in a 16×16 torus.

ures 9 and 10 show that the superiority of the dimension-exchange method over the diffusion method holds under the synchronous invocation policies as well, and that the ADE algorithm has an advantage over the diffusion method in both short and long terms.

7 Conclusions

In this paper, we made a comparison between two classes of nearest neighbor load balancing algorithms, the dimension-exchange (DE) and the diffusion (DF) methods, with respect to their efficiency in driving any initial workload distribution to a uniform distribution and their ability in controlling the growth of the variance among the processors' workloads. We focused on their four instances, the ADE, the ODE, the ADF and the ODF, which are the most common versions in practice. The comparison was made comprehensively in both one-port and all-port communication models with consideration of various implementation strategies: synchronous/asynchronous invocation policies and static/dynamic random workload behaviors.

Let " $a \succ b$ " denote the relationship that a outperforms b , and " $a \sim b$ " the relationship that a is approximately equivalent to b in performance. Then, our comparative results can be summarized as in Tables 3 and 4.

Specifically, we showed that the dimension-exchange method outperforms the diffusion

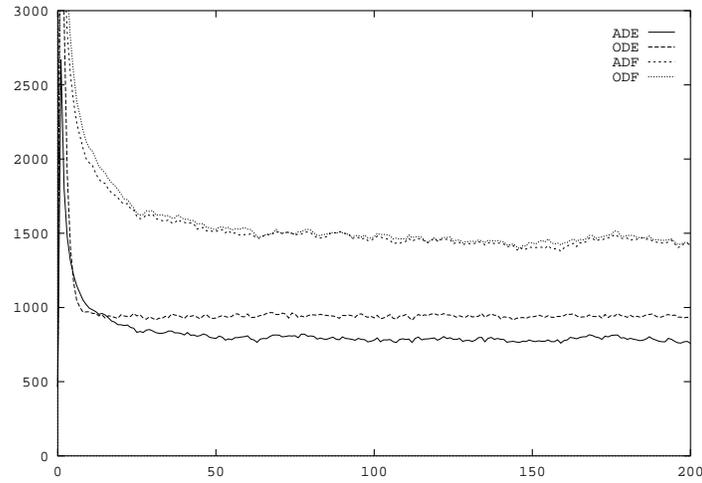


Figure 10: Change of the system workload variance during a dynamic synchronous load balancing process in a 16×16 mesh.

	Static load balancing	Dynamic load balancing
Synchronous	$ODE \succ ADE \succ ODF \sim ADF$	$ADE \succ ADF$ $ADE \succ ODE$ $ADF \succ ODF$
Asynchronous	$ADE \succ \{ADF, ODF\}$ $ADF \succ ODF$ in case $n = 1$ $ODF \succ ADF$ in case $n \geq 2$	same as left

Table 3: Summary of comparative results in the one-port communication model in n -D meshes and n -D tori.

	Static load balancing	Dynamic load balancing
Synchronous	$ODE \succ ADE \sim ODF \sim ADF$	$ADE \succ ADF$ $ADE \succ ODE$ $ADF \succ ODF$
Asynchronous	$\{ADF, ODF\} \succ ADE \succ ODE$ $ADF \succ ODF$ in case $n = 1$ $ODF \succ ADF$ in case $n \geq 2$	same as left

Table 4: Summary of comparative results in the all-port communication model in n -D meshes and n -D tori.

method in the one-port communication model. In particular, the ODE algorithm lends itself best to synchronous implementation in the static situation. We also revealed the superiority of the dimension-exchange method in synchronous load balancing even in the all-port communication model. The strength of the diffusion method is in asynchronous implementation in the all-port communication model. The ODF algorithm performs best in that case.

The comparative study not only provides an insight into nearest neighbor load balancing algorithms, but also offers practical guidelines to system developers in designing load balancing architectures for various parallel computational paradigms. We applied both the diffusion and the dimension-exchange methods in distributed branch-and-bound computations, and partly verified our comparative results in both static and dynamic asynchronous implementations in the platforms of Parsytec GCPP (PowerPC-based) and Parsytec GCel (Transputer-based) multicomputers [17]. We also evaluated their synchronous performances in real applications in periodic re-mapping of data parallel computations in [19].

Acknowledgments

We are grateful to H. L. Xie for his careful proofreading and the anonymous referees for their valuable comments.

References

- [1] I. Ahmad, A. Ghafoor, and K Mehrotra. Performance prediction for distributed load balancing on multicomputer systems. In *Proceedings of Supercomputing'1991*, pages 830–839 (1991).
- [2] L. V. Kale. Comparing the performance of two dynamic load distribution methods. In *Proceedings of International Conference on Parallel Processing*, pages 8–12 (1988).
- [3] V. Kumar, A. Y. Grama, and N. R. Vempaty. Scalable load balancing techniques for parallel computers. *Journal of Parallel and Distributed Computing*, 22(1):60–79 (1994).
- [4] M. Willebeek-LeMair and A. P. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, 4(9):979–993 (1993).
- [5] C.-Z. Xu and F.C.M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. *Journal of Parallel and Distributed Computing*, 16(4):385–393 (1992).
- [6] C.-Z. Xu and F.C.M. Lau. Iterative dynamic load balancing in multicomputers. *Journal of Operational Research Society*, 45(7):786–796 (1994).
- [7] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice-Hall Inc. (1989).
- [8] G. Cybenko. Load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301 (1989).

- [9] C.-Z. Xu and F.C.M. Lau. The generalized dimension exchange method for load balancing in k -ary n -cubes and variants. *Journal of Parallel and Distributed Computing*, 24(1):72–85 (1995).
- [10] S. L. Johnsson and C.-T. Ho. Spanning graphs for optimum broadcasting and personalized communication in hypercubes. *IEEE Transactions on Computers*, 38(9):1249–1268 (1989).
- [11] D. W. Krumme, G. Cybenko, and K. N. Venkataraman. Gossiping in minimal time. *SIAM Journal on Computing*, 21(1):111–139 (1992).
- [12] J. B. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Practice and Experience*, 2(4):289–313 (1990).
- [13] J.-W. Hong, X.-N. Tan, and M. Chen. From local to global: an analysis of nearest neighbor balancing on hypercube. In *Proceedings of ACM-SIGMETRICS*, pages 73–82 (1988).
- [14] X.-S. Qian and Q. Yang. Load balancing on generalized hypercube and mesh multiprocessors with lal. In *Proceedings of 11th International Conference on Distributed Computing Systems*, pages 402–409 (1991).
- [15] C.-Z. Xu and F.C.M. Lau. Optimal parameters for load balancing with the diffusion method in mesh networks. *Parallel Processing Letters*, 4(2):139–147 (1994).
- [16] S. H. Hosseini, B. Litow, M. Malkawi, J. Mcpherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *Journal of Parallel and Distributed Computing*, 10:160–166 (1990).
- [17] C.-Z. Xu, S. Tschoeke, and B. Monien. Performance evaluation of load distribution strategies in parallel branch-and-bound computations. Technical report, Dept. of Electrical and Computer Engg., Wayne State University (1995).
- [18] R. Diekmann, D. Meyer, and B. Monien. Parallel decomposition of unstructured FEM-meshes. Technical report, Dept. of Mathematics and Computer Science, University of Paderborn, Germany (1995).
- [19] C.-Z. Xu and F.C.M. Lau. Decentralized remapping of data-parallel computations with the generalized dimension exchange method. In *Proceedings of 1994 Scalable High Performance Computing Conference*, pages 414–421. IEEE Computer Society Press (1994).
- [20] J. Song. A partially asynchronous and iterative algorithm for distributed load balancing. *Parallel Computing*, 20(6):853–868 (1994).
- [21] R. Lüling and B. Monien. A dynamic distributed load balancing algorithm with provable good performance. In *Proceedings of 5th ACM Symposium on Parallel Algorithms and Architectures*, pages 164–172 (1993).
- [22] W. J. Dally. Performance analysis of k -ary n -cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–785 (1990).

-
- [23] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, 26:62–76 (1993).
 - [24] G. Ramanathan and J. Oren. Survey of commercial parallel machines. *ACM Computer Architecture News*, 21(3):13–33 (1993).
 - [25] D. M. Nicol and J. H. Saltz. Dynamic remapping of parallel computations with varying resource demands. *IEEE Transactions on Computers*, 37(9):1073–1087 (1988).
 - [26] S. Ranka, Y. Won, and S. Sahni. Programming a hypercube multicomputer. *IEEE Software*, 5:69–77 (1988).
 - [27] Y. Shih and J. Fier. Hypercube systems and key applications. In K. Hwang and D. De-groot, editors, *Parallel Processing for Supercomputers and Artificial Intelligence*, pages 203–243. McGraw-Hill Publishing Co. (1989).
 - [28] S. Fiorini and R. J. Wilson. Edge-coloring of graphs. In L. W. Beineke and R. J. Wilson, editors, *Selected Topics in Graph Theory*, pages 103–125. Academic Press (1978).
 - [29] B. Ghosh and S. Muthukrishnan. Dynamic load balancing in distributed networks by random matchings. In *Proceedings of 6th ACM Symposium on Parallel Algorithms and Architectures* (1994).
 - [30] A. N. Choudhary, B. Narahari, and R. Krishnamurti. An efficient heuristic scheme for dynamic remapping of parallel computations. *Parallel Computing*, 19:621–632 (1993).
 - [31] J. De Keyser and D. Roose. Load balancing data parallel programs on distributed memory computers. *Parallel Computing*, 19:1199–1219 (1993).