Bounds on the Speedup and Efficiency of Partial Synchronization in Parallel Processing Systems

C.S. Chang and R. Nelson
IBM Research Division
T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

(November, 1990; revision: November, 1993)

Abstract

In this paper, we derive bounds on the speed-up and efficiency of applications that schedule tasks on a set of parallel processors. We assume that the application runs an algorithm that consists of N iterations and before starting its $i+1$'st iteration, a processor must wait for data (i.e., synchronize) calculated in the i'th iteration by a subset of the other processors of the system. Processing times and interconnections between iterations are modeled by random variables with possibly deterministic distributions. Scientific applications consisting of iterations of recursive equations are examples of applications that can be modeled within this formulation. We consider the efficiency of such applications and show that, although efficiency decreases with an increase in the number of processors, it has a nonzero limit when the number of processors increases to infinity. We obtain a lower bound for the efficiency by solving a equation which depends on the distribution of task service times and the expected number of tasks needed to be synchronized. We also show that the lower bound is approached if the topology of the processor graph is "spread-out", a notion we define in the paper.

1 Introduction

Multiple processor systems offer the promise of both increasing the throughput of a computing system as well as decreasing job response time through the use of parallel processing. In such systems, throughput is increased by the addition of more processors and, associated with this increased capacity, is a reduction in job response time due to decreased queueing delays. Further reduction in job response time requires exploiting parallelism within the job by simultaneously executing a job's tasks on multiple processors. As initially pointed out by Amdahl [1] there are fundamental limitations on speedups (and thus also on the efficiency) that are obtainable through parallel execution. Simple, deterministic, models of parallel execution [1, 20, 25] show that the amount of parallelism and vectorization found in the workload place bounds on possible speedup, and that large speedups are only achievable for workloads having large degrees of parallelism. In particular, although adding processors offers the potential of linear speedup, very few applications actually achieve such efficiencies. Exceptions to this occur in applications, as pointed out in [24], that have the property that the amount of parallelism found in the job scales as a function of the problem size. Large speedups are also achievable by having a multiprogramming level greater than one, and thus processors not utilized by one job can be used by other jobs in the multiprogramming set [33].

Even under the ideal assumptions that jobs have unbounded parallelism, costs associated with managing the cooperation between a job's tasks (i.e., synchronization, data sharing, etc...) put limitations on possible speedups. The fact that synchronization costs can have a major influence on the performance, leads Stone [38] to propose measuring machine performance in terms of MSYPS (millions of synchronizations per second) rather than in the traditional MIPS (millions of instructions per second). In [14, 21] deterministic, computation models that assume unbounded parallelism are analyzed to derive optimal task partitioning among processors. These models show that, since synchronization costs grow with the problem size, even with unbounded job parallelism there is a limitation on the number of processors that should be used in a computation and thus that speedups are not linear. The costs that arise in a computing system associated with synchronization depend upon the underlying software or hardware mechanisms used [3, 18, 23, 34] and typically grow linearly or logarithmically (e.g., a butterfly type algorithm) with the number of synchronizing processors. An additional cost of synchronization arises from randomness in task execution times. These variations can arise either from resource sharing, such as memory conflicts, arise from the inherent randomness of a computation, or from both effects. These random variations result in staggering task completion times and have different effects depending on the parallel processing architecture [4, 31, 32]. An analysis of how different interconnection structures influence the efficiency of several scientific applications can be found in [22].

In this paper we focus on this last effect, randomness in task execution time, to determine fundamental limitations of task synchronization. Our model assumes that all system costs associated with synchronization are included in the task execution times. This assumption is less restrictive, as in our model, if task completion times are staggered due to random execution times. In these cases, the synchronization time is overlapped with task execution which tends to reduce the effect of overhead due to resource sharing. We assume that K processors execute an iterative algorithm and that a given processor cannot proceed to the $i+1$'st iteration until a subset of the processors to which it is connected all complete their i'th iteration. If this subset of processors is a proper subset of all the processors, then a *partial* synchronization is said to occur [11, 12, 23]. If the subset consists of all the processors than the synchronization is a *barrier* synchronization. A performance model of an iterative algorithm executed on different architectures using barrier synchronization can be found in [17] and of partial synchronization in [12]. Barrier synchronization typically is required when the value of a global variable is needed in successive iterations. Partial synchronization can occur, for example, in a domain decomposition method where interactions are necessary only between neighboring subdomains.

Our interest in this paper is to determine how the type of synchronization affects the average utilization of the processors (commonly called the efficiency of the system [11, 19]). Clearly barrier synchronization is the most deleterious and an argument presented in Section 2 shows that if task execution times are not bounded, then processors have a limiting utilization of zero as their number grows without bound. A natural question to ask is whether, under similar assumptions, this zero limit also exists for partial synchronization. Since many scientific applications, such as iterative, multigrid or domain decomposition methods use partial synchronization, (see [35] for PDE solutions and also [15] for a general reference) the answer to this question is tantamount to determining if there is a fundamental limitation on the efficiency that can be expected from such applications. Our results show, under general assumptions, that the asymptotic efficiency of partial synchronization is bounded away from zero and we derive an equation that can be solved to determine this bound. The bound requires that task execution times have a moment generating function, i.e., that all moments are finite, and thus is not a restriction for practical applications. In barrier synchronization, since all processors start an iteration at the same time, efficiency calculations require determining statistical properties of the maximum of independent random variables. One of the main difficulties of analyzing partial synchronization schemes is that processors start iterations at random times that are highly correlated. Capturing these dependencies in an analytic model is one of the contributions of this paper.

One of the problems of executing an iterative algorithm on multiple processors is *mapping* the tasks composing the problem onto the processors of the system. For a given interconnection architecture, one typically attempts to create a mapping that minimizes communication delays (see [8, 28] for instance). The idea here is that a poor mapping results in excessive communication delay which can lead to poor performance. One example where this is the case is when the computation time is nearly deterministic and is roughly of the same order as the time needed to communicate between processors. Our work points out another aspect of the mapping that becomes increasingly important in cases where the computation time of processes is variable and is large relative to communication times. For these cases, another feature of the mapping, i.e., how the resultant processor task graph *spreads out*, is an important feature of the mapping and strongly influences the efficiency of the mapping. Our results, which are independent of the particular interconnection architecture, depend only on the logical structure of the mapping of work onto processors and we require only that execution times have a moment generating function, i.e., that all moments are finite. Our results show that an additional feature, besides communication delays, that should be considered in creating mappings for iterative algorithms, is the way the task graph spreads out.

In Section 2 of this paper we present our models for the computation and architecture of the system we study. We assume that an iterative algorithm has been mapped to a processor architecture. The resulting computation can be viewed as a random acyclic graph with a leveled structure. The interconnection structure between the levels of this graph is determined by mapping the algorithm onto the architecture. The efficiency of the resultant execution model depends upon the way in which the processor graph "spreads out", a notion we precisely characterize. We first derive the result that barrier synchronization has a limiting efficiency of zero and then develop bounds for the special case where tasks execution times are exponentially distributed. Guided by intuition gained from this special case, we derive a bound for the efficiency for the general case of Section 3. Random graphs that achieve the lower bound are then created in Section 4. Throughout this section we attempt to provide intuition for our results and proofs for the theorems are given in the appendix. Section 5 presents our results for different topologies and we conclude in Section 6.

2 The Model

In Section 2.1 we describe the computational model that arises from executing an iterative algorithm on multiple processors. This requires that we specify how work is mapped onto the processors of the system and we do this using graph theoretic methods leading to a random task graph model. Preliminary results for this model are found in Section 2.2. In Section 2.3 we present results for a special task graph model corresponding to computation where processor interdependencies are identical for all iterations and task service times are exponentially distributed. This special case is used to motivate our general results that are found in Section 3.

2.1 The Random Task Graph Model

In this section we consider how one maps work from an iterative algorithm onto the processors of the system. We suppose we have an iterative algorithm consisting of N iterations executed on K processors. For each iteration, let the individual units of work, called *tasks* be denoted by $w_\ell^n, \ell = 1, 2, \ldots, L, n = 1, 2, \ldots, N$. Tasks are assumed to have random execution times and have certain precedence relationships between successive iterations. For the n'th iteration, each task w_ℓ^n, $\ell = 1, 2, \ldots, L$, is a assigned to a processor. If $L \leq K$ then all tasks can be assigned to a unique processor that is dedicated solely to its execution. If $L > K$, however, then multiple tasks from the n'th iteration will be assigned to the same processor. The mapping of tasks onto processors induces a graph that describes precedence relationships between processors between successive iterations. In this graph, there is an arc between processor k at iteration n to processor k' at iteration $n+1$ if processor k' executes any task at iteration $n+1$ which requires results from any task executed at iteration n by processor k.

Processors which have precedence relationships with processor k at iteration $n+1$ are said to be *neighbors* of processor k at iteration $n+1$. We assume that processor k can only begin work on its $n+1$'st iteration after it, and all of its neighbors at iteration $n+1$, have finished work on their n'th iterations. In some systems, this restriction is relaxed. In particular if, during the computation, it can be determined that some portion of the work for processor k at iteration $n+1$ at a given time t, does not depend upon results from neighboring processors past time t, then processor k can begin execution of that portion of work. We do not model such complexities in our analysis.

The resultant computation can be best modeled by a directed, acyclic, *random task graph* where directed arcs imply precedence. Nodes in this graph, $(n, k), 1 \leq n \leq N, 1 \leq k \leq K$, correspond to the work allocated to processor k on its n'th iteration. The total amount of work assigned to node (n, k) is a random variable denoted by $X_{n,k}$, $1 \leq n \leq N$, $1 \leq k \leq K$. (Processors, at the start of the algorithm, are assumed to be loaded with initial work used for the first iteration.) Let the indicator random variable $L_n(i, j), 1 \leq n \leq N-1, 1 \leq i, j \leq K$, represent the existence of an arc from node (n, i) to $(n+1, j)$ (i.e., $L_n(i, j) = 1$ if there is an arc and 0 otherwise). No arc exists between nodes (m, i) and (n, j) if $m \neq n-1$. In this graph model, barrier synchronization is modeled with $L_n(i, j) = 1, 1 \leq n \leq N-1, 1 \leq i, j \leq K$. In all other cases, i.e., if for every j there exists an i such that for a given n, $L_n(i, j) = 0$ then partial synchronization is modeled. We note that a graph with deterministic arcs (i.e., one corresponding to computation where the data dependencies between nodes does not differ in different iterations) can be modeled by specifying the value of $L_1(i, j)$ and then setting $L_n(i, j) = L_1(i, j), n = 2, 3, \ldots, N-1$. To avoid repetitive language, we sometimes refer to the task graph associated with an iterative algorithm as a *loop*.

We call a loop *lateral* if processor k always depends on data calculated in its previous iteration, i.e., if $L_n(k,k) = 1, 1 \le n \le N-1, 1 \le k \le K$. A loop is said to be *independent* and *identically distributed* (i.i.d.) if $\{X_{n,k}\}_{n,k}$ are i.i.d. random variables and independent of $\{L_n(i,j)\}_{n,i,j}$ and the random vectors $\{(L_n(i,j), i = 1, \ldots, K, j = 1, \ldots, K)\}_n$ are also i.i.d. random variables. Without loss of generality, for i.i.d. loops we assume that $EX_{n,k} = 1$. Note that the arcs connecting iterations n and $n+1$ in i.i.d. loops are statistically identical for all $n, 1 \le n \le N-1$ (thus graphs with deterministic arcs are i.i.d. loops). A realization of an i.i.d. lateral loop is shown in figure 1.

For i.i.d. task graphs, we define $r_i, 1 \le i \le K$, to be the *expected outdegree* of node i where

$$r_i \stackrel{\text{def}}{=} E \sum_{j=1}^{K} L_1(i,j). \tag{1}$$

The completion time of the n-th iteration on processor k, denoted by $B_{n,k}$, is defined by the following recursive equation:

$$B_{1,k} \stackrel{\text{def}}{=} X_{1,k} \tag{2}$$

$$B_{n,k} \stackrel{\text{def}}{=} X_{n,k} + \max_{\{i : L_{n-1}(i,k)=1\}} \{B_{n-1,i}\}. \tag{3}$$

The completion time of the n-th iteration, denoted by B_n, is defined as $\max_k \{B_{n,k}\}$ and the completion time of a loop with N iterations on K processors, denoted by $T(N,K)$, is defined as B_N.

We define the *speed up* of a loop with N iterations on K processors, denoted by $S_p(N,K)$, to be the ratio of the expected completion time of the loop on a single processor to that on K processors i.e.,

$$S_p(N,K) \stackrel{\text{def}}{=} \frac{E\sum_{n,k} X_{n,k}}{ET(N,K)} = \frac{NK}{ET(N,K)}.$$

The *efficiency* of a loop with N iterations on K processors, $\rho(N,K)$, is defined to be

$$\rho(N,K) \stackrel{\text{def}}{=} \frac{S_p(N,K)}{K}. \tag{4}$$

These are standard definitions (see [1, 3, 10, 19, 20, 24]). We note that the efficiency, $\rho(N,K)$ is equivalent to the average processor utilization during the computation.

The *asymptotic efficiency* of an i.i.d. loop with N iterations on K processors is defined to be the constant $\rho(K), 0 \le \rho(K) \le 1$, given by

$$\lim_{N\to\infty} \rho(N,K) = \lim_{N\to\infty} \frac{N}{ET(N,K)} = \rho(K), \tag{5}$$

provided the limit exists.

2.2 Preliminary Results

We now establish some properties of the random task graphs defined above.

First we establish simple bounds for the completion time of a loop with N iterations for lateral loops (see [36] for another version of this result).

Proposition 2.1 *If a loop is lateral, it then follows from (3) that*

$$\max_k \{\sum_{n=1}^{N} X_{n,k}\} \leq T(N,K) \leq \sum_{n=1}^{N} \max_k \{X_{n,k}\}. \tag{6}$$

Moreover, $T(N,K)$ *is the longest path among all the paths through* $(1,k_1),(2,k_2),\ldots,(N,k_N)$*, i.e.,*

$$T(N,K) = \max_{k_1,\ldots,k_N} \{(\prod_{n=2}^{N} L_{n-1}(k_{n-1},k_n)) \sum_{n=1}^{N} X_{n,k_n}\}. \tag{7}$$

Next we show that the limit defining the asymptotic efficiency of a lateral i.i.d. loop always exists.

Theorem 2.2 *In a lateral i.i.d. loop with* N *iterations on* K *processors the limit defining* $\rho(K)$ *in equation (5) exists. Moreover,* $1/\rho(K) = \lim_{N\to\infty} T(N,K)/N = \inf_N ET(N,K)/N$ *a.s.*

Although our definition of $\rho(K)$ is based on the limit of $ET(N,K)/N$, Theorem 2.2 shows that our definition is equivalent to the definition based on the limit of $T(N,K)/N$ (for almost every sample path).

Proof: Let $T_{m,n}$ be defined as

$$T_{m,n} = \max_{k_m,\ldots,k_n} \{ \prod_{s=m+1}^{n} L_{s-1}(k_{s-1},k_s) \sum_{s=m}^{n} X_{s,k_s}\} \tag{8}$$

and thus $T(N,K) = T_{1,N}$ and corresponds to the maximum length path from the beginning to the end of the task graph. Consider dividing a task graph of n iterations into two disjoint segments. The first segment corresponds to iterations 1 to m and the second from iterations $m+1$ to N. Clearly the sum of the maximal length paths within these segments cannot be smaller than the maximal length path in the original task graph and thus $T_{1,n} \leq T_{1,m} + T_{m+1,n}$. Moreover, one observes that

(i) the joint distribution of the process, $\{T_{m,n}\}$, is the same as that of $\{T_{m+1,n+1}\}$,

(ii) for any increasing sequence $(n_1, n_2, \ldots, n_l)$, $T_{n_i, n_{i+1}-1}$ are independent,

(iii) $ET_{1,n} \leq nK < \infty$ for $n < \infty$.

Thus, $\{T_{m,n}\}$ form an independent nonnegative subadditive process. By Kesten's theorem (see Kingman [27]), we have

$$\lim_{n\to\infty} \frac{T_{1,n}}{n} = \lim_{n\to\infty} \frac{ET_{1,n}}{n} = \inf_n \frac{ET_{1,n}}{n} = 1/\rho(K) \;\; a.s. \tag{9}$$

■

We note that the condition for $X_{n,k}$ in Theorem 2.2 can be relaxed from i.i.d. random variables to stationary ergodic sequences (see [5] for details). In the next proposition we show that the asymptotic efficiency of barrier synchronization has a limiting value of zero as $K \to \infty$.

Proposition 2.3 *If a loop is i.i.d. with barrier synchronization and if $P[X_{1,1} > t] > 0$, all $t > 0$ then $\rho(K) \to 0$ as $K \to \infty$.*

The proof follows from the fact that for barrier synchronization

$$\rho(K) = \frac{1}{E \max\{X_{1,1}, X_{1,2}, \ldots, X_{1,K}\}} \tag{10}$$

and that the expected value of the maximum grows to infinity as $K \to \infty$ [16].

2.3 Deterministic Task Graphs with Exponential Task Times

In this section we consider a collection of lateral i.i.d loops on K processors with deterministic arcs. The task service times are exponentially distributed with mean 1 (see [31, 32] for analysis of this model for different architectures). Since the expectation of the maximum of K independent and exponentially distributed random variables with mean 1 is the harmonic series, $H(K) = \sum_{k=1}^{K} 1/k$, it then follows from (6) that

$$\frac{1}{H(K)} \leq \rho(K) \leq 1. \tag{11}$$

In the following theorem, we show that the bounds can be further tightened by worst case analysis.

Theorem 2.4 *Consider a deterministic lateral loop with expected outdegrees satisfying*

$$r_l \leq r_k \leq r_u, 1 \leq k \leq K. \tag{12}$$

Then the asymptotic efficiency of such a graph with exponential task times is bounded by

$$\frac{1}{H(\lceil r_u H(\lceil \frac{K}{r_u} \rceil) \rceil)} \leq \rho(K) \leq \frac{1}{H(r_l)} \tag{13}$$

where $\lceil x \rceil$ *is the smallest integer that is larger than* x.

Proof: Suppose processors complete their n-th iterations in the order $n_1, \ldots, n_K$. Let W_n be the set of processors that have not finished their $(n+1)$'st iteration at the time n_K finishes its n'th iteration. Let $M_n = |W_n|$ be the cardinality of W_n. It is clear that

$$T(n+1, K) = T(n, K) + \max_{i \in W_n} \{X^R_{n+1,i}\} \tag{14}$$

where $X^R_{n+1,i}$ are random variables corresponding to the remaining task execution times at the time n_K finishes its n-th iteration. By the memoryless property of exponential random variables, $X^R_{n+1,i}$ has the same distribution as $X_{n+1,i}$. Taking expectations and simplifying shows that

$$ET(N, K) = \sum_{n=0}^{N-1} EH(M_n) \tag{15}$$

and the bounds are obtained by determining bounds on $EH(M_n)$. Since we will only concentrate on the n-th iteration, the subscript n in M_n will be omitted throughout to simplify the notation.

(**upper bound**) We make optimistic assumptions and assume that the only processors that have not finished their $n+1$'st iterations at the time n_K finishes its n iteration are the processors with incoming arcs from n_K. By assumption, the outdegree of n_K is at least r_l. Using this implies that $EH(M) \geq H(r_l)$.

(**lower bound**) Let $\mathcal{A} = \{j | K - \lceil \frac{K}{r_u} \rceil < j \leq K\}$ and define $\mathcal{S}_j$ to be the set of processors with incoming arcs from $n_j, j \in \mathcal{A}$. The set $\mathcal{S}_j$ is called the neighbors of n_j. We make pessimistic assumptions and assume that processors $n_K, n_{K-1}, \ldots, n_{K - \lceil \frac{K-1}{r_u} \rceil}$ have disjoint sets of neighbors in the next iteration. Furthermore we assume that they are maximally connected to r_u other nodes (except $n_{K-\lceil \frac{K}{r_u} \rceil}$) and thus that they hold as many as processors as is possible. We observe that the cardinality of these sets satisfies $|\mathcal{S}_j| \leq r_u$. By assumption,

$$\cup_{j\in\mathcal{A}} \mathcal{S}_j = \{1, 2, \ldots, K\} \tag{16}$$

and thus all the processors of the $n+1$'st iteration are accounted for in the sets $\mathcal{S}_j$. We define $Z_j, j \in \mathcal{A}$, to be the number of processors in $\mathcal{S}_j$ that have not completed their $n+1$'st iteration when n_K finishes its n'th iteration and let the total number be given by $Z = \sum_{j\in\mathcal{A}} Z_j$. The probability that any node $k_j \in \mathcal{S}_j$ has not completed its $n+1$'st iteration when n_K finishes its n'th is smaller than the probability that X_{n+1,k_j}, $k_j \in \mathcal{S}_j$, is longer than $\max(\tilde{X}^R_{n,n_{j+1}}, \ldots, \tilde{X}^R_{n,n_K})$, where $\tilde{X}^R_{n,k}$ is the remaining processing time of the task (n,k) when the processor n_j finishes its n-th iteration. By the memoryless property of exponential random variables and the assumption that the task service times are identically distributed, it follows that this probability is $\frac{1}{K-j+1}$ so that $E[Z_j] \leq r_u/(K-j+1)$. Since $Z = \sum_{j\in\mathcal{A}} Z_j$ it follows that

$$EZ \leq \sum_{j\in\mathcal{A}} \frac{r_u}{K-j+1} \leq r_u H(\lceil \frac{K}{r_u} \rceil) \leq \lceil r_u H(\lceil \frac{K}{r_u} \rceil) \rceil. \tag{17}$$

Since $H(k)$ is increasing concave, it then follows from Jensen's inequality and $M \leq Z$ that $EH(M) \leq H(EM) \leq H(EZ)$. By (17), we conclude the proof for the lower bound. ■

We note that the lower bound in Theorem 2.4 can be tightened by defining a linear interpolation function $\tilde{H}(x)$ of $H(k)$, i.e.,

$$\tilde{H}(x) \stackrel{\text{def}}{=} (\lfloor x \rfloor + 1 - x) H(\lfloor x \rfloor) + (x - \lfloor x \rfloor) H(\lfloor x \rfloor + 1), \tag{18}$$

where $\lfloor x \rfloor$ is the largest integer that is smaller than x. Moreover, $\sum_{j\in\mathcal{A}} r_u/(K-j+1)$ in (17) can be calculated exactly as

$$z \stackrel{\text{def}}{=} r_u H(\lfloor \frac{K}{r_u} \rfloor) + \frac{(K - \lfloor \frac{K}{r_u} \rfloor r_u)}{(\lceil \frac{K}{r_u} \rceil)}.$$

The new lower bound is $1/\tilde{H}(z)$. We expect that the lower bound is tight when $\frac{K}{r_u}$ is small as we will show in Section 5.

One can relax the requirement that all task execution times have the same mean in Theorem 2.4 by making worst case assumptions. We state this fact in the following Corollary.

Corollary 2.5 *Suppose the exponential service rates of tasks, μ_k, satisfies*

$$\mu_l^{-1} \leq \mu_k^{-1} \leq \mu_u^{-1}, 1 \leq k \leq K. \tag{19}$$

Then the asymptotic efficiency is bounded by

$$\frac{1}{\mu_u H(\lceil r_u H(\lceil \frac{K}{r_u} \rceil) \rceil)} \leq \rho(K) \leq \frac{1}{\mu_l H(r_l)} \tag{20}$$

We also note that (3) only contains the maximum function and summation of independent random variables. It is easy to establish the variability (convex) ordering (recall that in Ross [37] or Stoyan [39] X_1 is less variable than ($\leq_{icx}$) X_2 if $Ef(X_1) \leq Ef(X_2)$ for all f increasing convex) for the efficiency among various distributions of task service times. Since the *New Better Than Used* (NBUE) distribution is less variable than the exponential distribution with the same mean, the lower bound in Theorem 2.4 is also a lower bound for loops whose task service times have NBUE distributions. We record this here as a corollary to Theorem 2.4.

Corollary 2.6 *The asymptotic efficiency of a deterministic lateral loop where task execution times are NBUE is lower bounded by the lower bound of equations (13).*

In barrier synchronization, Proposition 2.3 showed that the asymptotic efficiency decreases to zero as K gets large. For an independent iteration loop where $r_k = 1, 1 \leq k \leq K$, it is clear that the asymptotic efficiency is equal to one. For any fixed expected outdegree $r > 1$, however, Theorem 2.4 shows that the asymptotic efficiency is at least of the order $\frac{1}{log(log(K))}$. Since this lower bound also decreases to zero when K approaches infinity it raises the question if this partial synchronization has an asymptotic efficiency of zero. If the answer to this question is yes, then parallel processing systems face a fundamental limitation in terms of speedup and efficiency when dependencies exist between successive iterations even if they are weak. Fortunately, as we show in the next section, there is a non-zero lower bound for partial synchronization if the moment generating function of task service times exists.

We end this section by observing that to prove the lower bound of Theorem 2.4 we assumed that the neighbors of the last finishing nodes formed a tree where between two iterations there are r_u arcs from nodes $n_K, n_{K-1}, \ldots, n_{K-\lceil \frac{K}{r_u} \rceil}$. In a loose sense, this "spreads out" the influence of the execution times of the n'th iteration of these nodes to as many other nodes in the system as possible for their $n+1$'st iteration. Larger efficiencies are obtained if the random task graph has a structure that tends to limit rather than diffuse the influence of task execution times. The notion of spreading out the influence of execution times is precisely defined in Section 4 where we use it to derive loops that achieve the general lower bound derived in Section 3. Not surprisingly these loops are obtained by constructing a random task graph that asymptotically contains an embedded tree. We note the lower bound generated here is sometimes tighter than that derived in Section 3 which considers more general task graphs.

3 A lower bound for lateral i.i.d. loops

In this section we derive a lower bound for the efficiency for lateral i.i.d. loops when task execution times have finite moments. Our results are summarized in the theorem below (similar bounds for stochastic petri nets are independently obtained by Baccelli and Liu [7] or [6], Section 8.3 and references therein).

Theorem 3.1 *(***Large Deviation Bound***) If the moment generating function of the task service times, $X_{n,k}$, exists for some $\theta_0 > 0$,*

$$\phi(\theta) \stackrel{\text{def}}{=} E e^{\theta X_{n,k}} < \infty \quad \textit{for } \theta \leq \theta_0 \tag{21}$$

and the expected outdegree is not larger than some constant r

$$E \sum_j L_n(i,j) \leq r, \quad \textit{for all } i \tag{22}$$

then the asymptotic efficiency of a lateral i.i.d. loop on K processors, $\rho(K)$, is bounded below by β_r, where

$$\frac{1}{\beta_r} = t^\star \stackrel{\text{def}}{=} \inf\{t \geq 1 : rm(t) < 1\} \tag{23}$$

and

$$m(t) \stackrel{\text{def}}{=} \inf_{\theta \geq 0} \{e^{-\theta t}\phi(\theta)\}. \tag{24}$$

The above theorem, which we will prove shortly, is an example of a large deviation result. The rate function for the process is defined as

$$I(t) \stackrel{\text{def}}{=} -\log(m(t)) = \sup_{\theta \geq 0}\{\theta t - \log\phi(\theta)\}, \tag{25}$$

and properties of $m(t)$ can be deduced from known properties of such rate functions. Specifically, it is known (see [40]) that $I(t)$ is continuous and convex and attains its unique minimal value at $t = EX = 1$. This implies that $m(t)$ is continuous and decreasing for $t \geq 1$. Except the trivial case that the task service times are deterministic and the same, β_r can be determined as the unique solution of $rm(t) = 1$.

It is also possible to simplify the solution for some problems when the task services times are sample means of i.i.d. random variables. To see this we prove the following lemma.

Lemma 3.2 *Let $\tilde{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Let $\tilde{\beta}_r$ (β_r) be the large deviation bound obtained in Theorem 3.1 for loops whose task service times have the same distributions as $\tilde{X}$ (X). Then*

$$\tilde{\beta}_r = \beta_{r^{\frac{1}{n}}}. \tag{26}$$

Proof: Let $\tilde{\phi}(\theta) \stackrel{\text{def}}{=} Ee^{\theta\tilde{X}}$ and $\tilde{m}(t) \stackrel{\text{def}}{=} \inf_\theta\{e^{-\theta t}\tilde{\phi}(\theta)\}$. Note that the minimum of $m(t)$ is attained at $\frac{\phi'(\theta)}{\phi(\theta)} = t$. Let θ_t be the θ that attains this minimum. Thus, $m(t) = e^{-\theta_t t}\phi(\theta_t)$. Since $\tilde{\phi}(\theta) = \phi(\frac{\theta}{n})^n$, the minimum of $\tilde{m}(t)$ is attained at $\frac{\phi'(\frac{\theta}{n})}{\phi(\frac{\theta}{n})} = t$. Similarly, let $\tilde{\theta}_t$ be the θ that attains this minimum and represent $\tilde{m}(t) = e^{-\tilde{\theta}_t t}\tilde{\phi}(\tilde{\theta}_t)$. Clearly, $\frac{\tilde{\theta}_t}{n} = \theta_t$ and thus $\tilde{m}(t) = m(t)^n$. Since β_r is the solution of $rm(t) = 1$ and $\tilde{\beta}_r$ is the solution of $r\tilde{m}(t) = 1$, we have $\tilde{\beta}_r = \beta_{r^{\frac{1}{n}}}$. ∎

As an example of this lemma, consider the case of Section 2.4 where task service times are exponential. Then $m(t)$ can be easily calculated as te^{1-t}. From Lemma 3.2, it follows that the large deviation bounds for loops whose task service times are of Erlang type j can be easily calculated from the solution of $r^{1/j}te^{1-t} = 1$. In figure 2 we show the large deviation bounds for the asymptotic efficiency, β_r, versus the expected degree, r, for loops whose task service times are of Erlang type j, $j = 1, \ldots 8$.

The rest of the section is devoted to the proof of this theorem. Our approach for the lower bound is adapted from Kingman [27].

Lemma 3.3 *Under the assumptions of Theorem 3.1, let $\mathcal{F}_n$ be the minimal σ-algebra generated by $\{L_{m-1}(i,j), i = 1, \ldots, K, j = 1, \ldots, K, m = 2, \ldots, n\}$ and $\{X_{m,k}, k = 1, \ldots, K, m = 1, \ldots, n\}$. Let*

$$M_n(\theta) \stackrel{\text{def}}{=} \frac{1}{(r\phi(\theta))^n} \sum_{k_1=1}^{K} \cdots \sum_{k_n=1}^{K} \left(\prod_{m=2}^{n} L_{m-1}(k_{m-1}, k_m)\right) e^{\theta\sum_{m=1}^{n} X_{m,k_m}}. \tag{27}$$

Then $\{M_n(\theta), \mathcal{F}_n, n \geq 1\}$ is a nonnegative supermatingale ($\prod_{m=2}^{1}(\cdots) \stackrel{\text{def}}{=} 1$ by convention) for $\theta \leq \theta_0$.

Proof: We need to verify that $E(M_{n+1}(\theta)|\mathcal{F}_n) \le M_n(\theta)$ a.s. Note that

$$E(M_{n+1}(\theta)|\mathcal{F}_n) = \frac{1}{(r\phi(\theta))^{n+1}} \sum_{k_1=1}^{K} \cdots \sum_{k_n=1}^{K} E\Big(\sum_{k_{n+1}=1}^{K} (\prod_{m=2}^{n+1} L_{m-1}(k_{m-1},k_m)) e^{\theta \sum_{m=1}^{n+1} X_{m,k_m}} \Big| \mathcal{F}_n \Big). \tag{28}$$

Since $L_{m-1}(i,j)$ and $X_{m,k}$, $m \le n$, are $\mathcal{F}_n$-measurable,

$$\begin{aligned} E & \Big(\sum_{k_{n+1}=1}^{K} (\prod_{m=2}^{n+1} L_{m-1}(k_{m-1},k_m)) e^{\theta \sum_{m=1}^{n+1} X_{m,k_m}} \Big| \mathcal{F}_n \Big) \\ = & \ (\prod_{m=2}^{n} L_{m-1}(k_{m-1},k_m)) e^{\theta \sum_{m=1}^{n} X_{m,k_m}} E(\sum_{k_{n+1}=1}^{K} L_n(k_n,k_{n+1}) e^{\theta X_{n+1,k_{n+1}}} | \mathcal{F}_n). \end{aligned} \tag{29}$$

By the definition of i.i.d. loops and (22), we have

$$\begin{aligned} E & \ (\sum_{k_{n+1}=1}^{K} L_n(k_n,k_{n+1}) e^{\theta X_{n+1,k_{n+1}}} | \mathcal{F}_n) \\ = & \ \sum_{k_{n+1}=1}^{K} EL_n(k_n,k_{n+1}) E e^{\theta X_{n+1,k_{n+1}}} \\ \le & \ r\phi(\theta) \end{aligned} \tag{30}$$

Inserting (30) and (29) into (28) completes the proof. ■

Lemma 3.4 *For* lateral *i.i.d. loops,*

$$\limsup_{N\to\infty} \frac{T(N,K)}{N} \le \frac{1}{\beta_r}, \quad a.s.$$

Proof: Let $Y_n(\theta) \stackrel{\text{def}}{=} (r\phi(\theta))^n M_n(\theta)$. It then follows from equation (7) of (Proposition 2.1) that

$$e^{\theta T(N,K)} = \max_{k_1,\ldots,k_N} \{ \prod_{n=2}^{N} L_{n-1}(k_{n-1},k_n) e^{\theta \sum_{n=1}^{N} X_{n,k_n}} \} \le Y_N(\theta) \tag{31}$$

if $\theta \ge 0$ where the inequality arises from the fact that the maximum of a set of non-negative random variables is less than their sum. Since $\{M_n(\theta), \mathcal{F}_n\}$ is a supermartingale, $EM_n(\theta) \le EM_1(\theta) = \frac{K}{r}$. Using this in (31) implies that

$$Ee^{\theta T(N,K)} \le \frac{K}{r}(r\phi(\theta))^N. \tag{32}$$

Using this in Chernoff's bound yields

$$P(\frac{T(N,K)}{N} > t) \leq e^{-\theta N t} E e^{\theta T(N,K)} \tag{33}$$

$$\leq \frac{K}{r}(e^{-\theta t} r \phi(\theta))^N \text{ for all } t. \tag{34}$$

Thus, $\sum_N P(\frac{T(N,K)}{N} > t) < \infty$ if $rm(t) < 1$. By Borel-Cantelli lemma, we have

$$\limsup_{N\to\infty} \frac{T(N,K)}{N} \leq t^*, \quad a.s. \tag{35}$$

Theorem 3.1 is an immediate consequence of Lemma 3.4. Theorem 3.1 can be generalized to the case where task execution times are not i.i.d. but rather only depend on the node and are independent of the iteration. Specifically let $X_{n,k}$ be the r.v. indicating the task service time for the k'th processor at the n-th iteration and let $\phi_{n,k}(\theta)$ be its moment generating function. Loosely speaking we bound the efficiency for this system by using a function $\psi(\theta)$ that is "not smaller" than any value of $\phi_{n,k}(\theta)$, for any θ. We then use $\psi(\theta)$ in Theorem 3.1. To precisely define this we must recall the definition of the essential supreme of a family of functions. For a given a set of functions, $\mathcal{F} \stackrel{\text{def}}{=} \{f_j, j = 1, 2, \ldots\}$, let $\mathcal{H}$ be the set of Borel-measurable functions that satisfy $h \geq f_j$, a.e. for all $f_j \in \mathcal{F}$. The essential supreme of $\mathcal{F}$ is a function $\tilde{h} \in \mathcal{H}$ that satisfies $\tilde{h} \leq h$, a.e. for all $h \in \mathcal{H}$. We denote this by $\tilde{h} = \text{esup}_j\{f_j\}$. We use this to state the following Corollary.

Corollary 3.5 *Let*

$$\psi(\theta) \stackrel{\text{def}}{=} esup_{n,k}\{\phi_{n,k}(\theta)\}. \tag{36}$$

Then the asymptotic efficiency of a lateral loop on K processors is bounded below by β_r where

$$\frac{1}{\beta_r} = \inf\{t \geq 1 : rm(t) < 1\} \tag{37}$$

where

$$m(t) \stackrel{\text{def}}{=} \inf_{\theta \geq 0}\{e^{-\theta t}\psi(\theta)\} \tag{38}$$

We note that if $X_{n,k} \leq_{icx} X$, then $\text{esup}_{n,k}\{\phi_k(\theta)\} \leq \phi(\theta) \stackrel{\text{def}}{=} Ee^{\theta X}$ since $e^{\theta x}$ is increasing convex in x if $\theta \geq 0$.

It is instructive to examine the lower bound for some special cases. First we note that the lower bound is independent of the number of processors K and depends only upon the value of r and the moment generating function of task service times. We first examine the lower bound for barrier synchronization under the assumptions of Proposition 2.3 by letting $r \to \infty$. The assumptions of the proposition imply that $m(t) > 0$ for any finite t since for any $\epsilon > t$

$$m(t) = \inf_{\theta \geq 0}\{e^{-\theta t}\phi(\theta)\} \geq \inf_{\theta \geq 0}\{e^{-\theta(t-\epsilon)}P(X > \epsilon)\} = P(X > \epsilon) > 0$$

by Chernoff's bound. Thus, for any finite t, there exists a r such that $rm(t) > 1$. It is clear then that the lower bound is zero as $r \to \infty$.

Consider now the case of unit deterministic service times. Here $m(t) = \inf_{\theta \geq 0} e^{-\theta(t-1)} = 0, t > 1$, and thus, for any r, $t = 1$ satisfies (23). The lower bound then is $\beta_r = 1$ as expected since all processors finish each iteration at the same time.

4 Loops that achieve the lower bound

In this section, we prove that Bernoulli i.i.d. loops (defined below) achieve the lower bound in the previous section. First we consider a collection of loops whose arcs (precedence constraints) can be generated by uniformly distributed random variables in the following manner. Let $\underline{p} = (p_1, p_2, \ldots, p_K)$ be a probability vector, i.e., $\sum_{j=1}^{K} p_j = 1$, and let $r > 0$ be a constant that will determine the expected outdegree of a node. For each iteration, node $k, 1 \leq k \leq K$, draws r samples according to $\underline{p}$ so that at each drawing an arc from node k to node j at the next iteration is drawn with probability $p_j, j = 1, 2, \ldots, K$. This procedure generates a random graph where the expected outdegree of each node is less than r (since we sample without replacement). This procedure is repeated for each iteration to generate the entire graph. We call the resultant graph, a *Bernoulli i.i.d. loop*. We construct a *Bernoulli lateral i.i.d. loop* by making the loop lateral and then distributing the remaining $r - 1$ arcs in the above fashion. Note that one example of such a loop is a lateral loop where remaining arcs between nodes are drawn uniformly with probability $1/K$.

Examples of computations that could result in a task graph with properties similar to Bernoulli i.i.d. loops include approximate factorization for sparse matrices where the arcs between iterations arise from fill created from the elimination algorithm or the simulation of particle systems with local interactions where a subset of particles are assigned to each processor. Another model where Bernoulli i.i.d. loops could arise was suggested to us by Vijay Naik [30]. Consider a reconfigurable interconnection network that is multiprogrammed with two jobs of different priority. Assume that the network is reconfigured according to the needs of the high priority job. The interconnections then for the low priority job could appear to vary randomly as a function of iteration. It would be surprising if such random changes increased the performance of the low priority work and our results show that such changes to the network decrease the efficiency of the lower priority job since Bernoulli i.i.d. loops achieve the lower bound of Section 3.

We now state theorems that establishes that such Bernoulli i.i.d. loops, under certain conditions of $\underline{p}$, achieve the lower bound of Section 3. Proofs for some of the results of this section are technical and can be found in the appendix. We are interested in two major issues regarding loops that achieve the lower bound. The first is determining sufficient conditions to guarantee that the lower bound is achieved. Similar to the lower bound construction of Theorem 2.4, this requires that the asymptotic task graph contains an embedded tree. We consider here Bernoulli i.i.d. task graphs (they can also be lateral) which have this property in appendix B. The second issue that we consider is the rate of convergence of task graphs to the lower bound. To understand this we must introduce the notion of majorization and establish a monotonic property (Theorem 4.2, proved in appendix A) that characterizes the rate of convergence for Bernoulli i.i.d. task graphs. Here, because of symmetry requirements, we must assume that the loops are not lateral.

In order to precisely define the notion of "spread out" mentioned previously we now introduce the concept of majorization [29].

Definition 4.1 *Let $\underline{p}^1$ and $\underline{p}^2$ be two K-dimensional vectors. $\underline{p}^1$ is majorized by $\underline{p}^2$ ($\underline{p}^1 \prec \underline{p}^2$) if*

(i) $\sum_{l=1}^{j} p_{[l]}^1 \leq \sum_{l=1}^{j} p_{[l]}^2$, $j = 1, \ldots, K-1$,

(ii) $\sum_{l=1}^{K} p_{[l]}^1 = \sum_{l=1}^{K} p_{[l]}^2$,

where $p_{[l]}^1$ ($p_{[l]}^2$) is the l-th largest element in $\underline{p}^1$ ($\underline{p}^2$).

Using this definition we say that distribution $\underline{p}^1$ is more spread out than distribution $\underline{p}^2$ if $\underline{p}^1$ is majorized by $\underline{p}^2$. Intuitively, there are two associated qualitative aspects to loops with precedence arcs that are more "spread-out" than those of other loops. First, the end points of precedence arcs are less concentrated onto particular processors. Second, the expected number of precedence arcs (after two or more iterations) is greater. Using this definition, we can state the first theorem of this section.

Theorem 4.2 *Let $T^1(N,K)$ ($T^2(N,K)$) be the completion time of the Bernoulli i.i.d. loop with parameters $\underline{p}^1$ ($\underline{p}^2$) and r. If $\underline{p}^1 \prec \underline{p}^2$, then $Ef(T^1(N,K)) \geq Ef(T^2(N,K))$ for all f increasing.*

The proof of this theorem is found in the appendix. In words the theorem says that the expectation of an increasing function of the completion time of a Bernoulli i.i.d. loop with N iterations and K processors is increased if the influence of task execution times is more spread out. Immediate results of this theorem are stated now as corollaries.

Corollary 4.3 *The expected completion time of Bernoulli i.i.d. loops on K processors is maximized when $\underline{p} = (\frac{1}{K}, \ldots, \frac{1}{K})$ since $\underline{p}$ is majorized by all other vectors of length K.*

Corollary 4.4 *The expected completion time of Bernoulli i.i.d. loops on K processors with parameters $(\frac{1}{K}, \ldots, \frac{1}{K})$ and r is increasing in K since $(\frac{1}{K+1}, \ldots, \frac{1}{K+1}, \frac{1}{K+1}) \prec (\frac{1}{K}, \ldots, \frac{1}{K}, 0)$.*

In corollary 4.4 note that the response time of a system with $K + 1$ processors with $(\frac{1}{K}, \ldots, \frac{1}{K}, 0)$ has a greater response time than on with K processors with $(\frac{1}{K}, \ldots, \frac{1}{K})$. Thus Corollary 4.4 implies that possible candidates for achieving the lower bound in the previous section are Bernoulli i.i.d. loops with "spread-out" arcs and on a large number of processors. We now state a theorem which establishes this result.

Theorem 4.5 *Let $\{G^{(K)}\}$ be a sequence of Bernoulli i.i.d. loops (they can also be lateral) on K processors with parameters $p_j^{(K)}$ and r. If*

$$\lim_{K \to \infty} p_j^{(K)} = 0 \tag{39}$$

for all $j = 1, \ldots, K$ then the asymptotic efficiency, $\rho(K)$, approaches the lower bound of Theorem 3.1 as K increases to infinity, i.e.,

$$\lim_{K \to \infty} \rho(K) = \frac{1}{\beta_r}. \tag{40}$$

A proof of Theorem 4.5 is found in the appendix. As an example of the conditions of Theorem 4.5 consider the case that $p_j^{(K)} = 1/K$. Clearly, $\lim_{K \to \infty} p_j^{(K)} = 0$.

Note that the lateral i.i.d. loops with deterministic arcs do not satisfy (39). Thus, the asymptotic efficiency for these loops with a large number of processors is not clear. In the next section, we present simulation results and propose experimental rules for these loops.

5 Simulation

In this section, we present the results of some simulation experiments we ran to determine the efficiency of finite execution graphs and compare these simulated values with the lower bound derived in Sections 2 and 3. We consider task graphs that correspond to nearest neighbor dependencies for interconnection networks that are rings, meshes and hypercubes. All networks are assumed to have wraparound connections and all task graphs are assumed to be lateral. Task service times are assumed to be exponentially distributed with mean 1. Associated with each network is a dimensional parameter denoted by $d, d \geq 1$ and a length parameter denoted by $L, L \geq 1$. The length parameter, L, is the number of nodes along any given dimension and the dimension parameter, d, depends upon the network topology. For rings, we assume that processors within a distance d of each other are connected and call such rings $d-$rings. For meshes and hypercubes, d is the dimension and we call such networks as $d-$meshes and $d-$cubes. We note that d-cubes are d-meshes with $L = 2$.

In Section 4 we introduced the notion of majorization to establish that the efficiency of Bernoulli i.i.d. task graphs decreased if the task graph became more "spread out". Here we provide another measure of this property for deterministic task graphs. For these graphs, let $\eta_n(k)$ be the number of *distinct* processors at the n'th iteration that can be reached along a directed path starting from processor k and let $\eta_n = \min_k \eta_n(k)$ (in all of the following examples $\eta_n(k) = \eta_n(j)$). The *diameter* of the task graph, denoted by η, is the minimum number of iterations, n such that $\eta_n = K$. At a given point in time, assume that processor k is at iteration i_k. It is clear that $\eta \geq \max_{j,k}\{|i_j - i_k|\}$ and thus that all processors are within η iterations of each other.

Consider now a processor k at its n'th iteration processing a very long task. If there is a path of length l from processor k to processor k', then k' can proceed at most to iteration $n + l$. Thus a long task can hold up all K processors to within a diameter of its iteration. Intuitively speaking, then, this implies that the efficiency of a task graph varies directly with its diameter. This is a complementary notion to the notion of spread-out introduced in Section 4. We can think of small diameter task graphs as being more spread out (for systems with the same number of processors).

Table 1 describes the properties of each of these networks for different values of d (these properties can be derived from simple recurrence relationships). The diameter of the task graph η and number of processors reached in two iterations, η_2, are also given in this table and will be used later in an approximation.

Table 1. Properties of Graphs

	Rings ($L \geq 2d+1$)	Meshes ($L > 2$)	Cubes
$r =$	$2d+1$	$2d+1$	$d+1$
$\eta =$	$\lceil (K-1)/2d \rceil$	$d\lceil (K^{1/d}-1)/2 \rceil$	$\lceil \log_2(K) \rceil$
$\eta_2 =$	$1+4d$	$1+2d+2d^2$	$(1/2)(2+d+d^2)$
$K =$	L	L^d	2^d

Generally, our results reveal two major influences on the efficiency of task graphs. The first of these, as outlined in Section 4, is that as interconnections become more spread out the efficiency decreases. The intuition behind this result follows directly from Theorem 4.2 and from our comments on the diameter above. Notice that for fixed K and d that rings have the largest diameter and cubes have the smallest. For similar configurations, then, we expect efficiencies to vary directly with the diameter, i.e., rings being the most efficient and cubes being the least. We now present details of our results.

We use the batch means method to generate confidence intervals for the simulations [26]. For each random task graph, we ran a single simulation and, after discarding the first 100 iterations, collect statistics in batches of 100 iterations. The simulation is halted when the width of the 95% confidence interval is within 0.01 of the calculated efficiency and the number of batches is at lease 10 for Bernoulli loops and meshes and at least 30 for rings and cubes.

Table 2. Bernoulli Loops

K	N	Simul	Large	Worst
10	10100	0.458	0.304	0.415
50	4100	0.388	0.304	0.339
100	1600	0.370	0.304	0.320
500	1100	0.343	0.304	0.290
1000	1100	0.340	0.304	0.281
5000	1100	0.328	0.304	0.265
10000	1100	0.325	0.304	0.259

We start by simulating Bernoulli i.i.d. lateral loops that have unit exponential task times on K processors with $r = 3$ and probability $1/K$ where we vary K, to determine the rate of convergence of these loops to the lower bound.

In table 2 we present the results from the simulation, from the large deviation bound of Theorem 3.1 (column *Large* in the table) and also the worst case bound of Theorem 2.4 (column *Worst* in the table). From Theorem 4.5 we know that $\lim_{K\to\infty} \rho(K) = \frac{1}{\beta_r}$ and, in figure 3, one sees that the convergence to this value is very slow. In figure 3, $r = 3$ and for $K = 10,000$ the simulated value is still 6 percent above the lower bound. The figure also shows that the worst case bound is very pessimistic, yielding at $K = 10,000$ a value that is 14 percent below the lower bound.

Table 3. $d-$Rings

d	r	η	η_2	K	N	Simul	Large	Worst	η_2 Approx
1	3	250	5	500	3100	0.420	0.304	0.290	0.438
2	5	125	9	500	3100	0.350	0.250	0.260	0.353
3	7	84	13	500	3100	0.316	0.226	0.243	0.314
4	9	64	17	500	3100	0.293	0.210	0.232	0.291
5	11	50	21	500	3100	0.277	0.200	0.224	0.274

In figure 4, present results from a simulation for d-ring loops for parameter values given in table 3. In the figure we plot the simulated efficiency, the large deviation bound in Theorem 3.1 and the worst case bound in Theorem 2.4. Additionally we plot an approximation for the efficiency which is described later in this section. It is clear from the figure that the large deviation bound and worst case bound are not tight, and the errors in these bounds are about 28 percent. The reason why the large deviation bound is not tight is that there is no embedded tree in the d-ring loop. The arcs which synchronize processors, in fact, are the least spread out in the sense defined in Section 4. Note that the worst case bound is tighter than the large deviation bound when d is large. This follows, as previously stated, from the fact that the number of processors is fixed. The worst case bound tends to be good when the ratio of the number of processors to the average degree is small since the loop in that case is similar to what happens in the worst case analysis. However, as $K \to \infty$ the worst case bound converges to zero while the large deviation bound remains constant.

We now explain the approximation used to generate the curve shown in figure 4. Recall that in the worst case derivation of Section 2.4 (Theorem 2.4), we calculated a bound for the number of processors that have not completed their $(n+1)$-th iteration when the last processor completes its n-th iteration. Our approximation, uses an estimate of this number. According to the upper bound in Theorem 2.4, this number is larger than the number of processors that can be reached in one iteration. Consider now the number of processors which can be reached in two iterations, denoted by η_2. Loosely speaking, η_2 is larger for computation graphs that are more spread out. Table 1 shows that η_2 increases when one considers rings, meshes and cubes for the same value of d. It is reasonable to believe that some measure of the amount that a task graph is spread out could be used to estimate the efficiency of a task graph and we propose using the value of η_2 for such an estimate. Specifically, we estimate the efficiency as the inverse of the expected value of the maximum of η_2 exponentials given by $\frac{1}{H(\eta_2)}$. We call this the η_2 *approximation.* The error between simulation and the estimated efficiency for rings, as seen in figure 4, is within 5 percent.

Table 4. $d-$Meshes

d	r	η_2	K	N	Simul	Large	Worst	η_2 Approx
2	5	13	$1,600$	1,100	0.317	0.250	0.247	0.314
3	7	25	$1,728$	1,600	0.274	0.226	0.230	0.262
4	9	41	$20,736$	1,100	0.247	0.210	0.204	0.232

In figure 5, we present results for meshes for parameter values given in table 4 (the diameters are not shown since the numbers of processors are different in the simulations). Since meshes have less locality than rings, the large deviation bounds are slightly tighter than those for rings. Moreover, the local preference decreases as the dimension of meshes increases. The errors in figure 5 decreases from 20 percent to 15 percent as the dimension increases. The worst case bound is not tight since the ratio of the number of processors to the degree is large. The η_2 approximation seems to be good even for meshes, having errors within 6 percent.

Table 5. $d-$cubes

d	r	η	η_2	K	N	Simul	Large	Worst	η_2 Approx
1	2	1	2	2	44,100	0.669	0.373	0.667	0.667
2	3	2	4	4	16,100	0.514	0.304	0.511	0.480
3	4	3	7	8	10,100	0.426	0.271	0.408	0.386
4	5	4	11	16	3,100	0.372	0.250	0.348	0.331
5	6	5	16	32	4,100	0.329	0.236	0.307	0.296
6	7	6	22	64	3,100	0.301	0.226	0.278	0.271
7	8	7	29	128	3,100	0.282	0.217	0.257	0.252
8	9	8	37	252	3,100	0.263	0.210	0.240	0.238
9	10	9	46	512	3,100	0.252	0.205	0.227	0.226
10	11	10	56	1024	3,100	0.241	0.200	0.217	0.217
11	12	11	67	2048	3,100	0.233	0.195	0.208	0.209
12	13	12	79	4096	3,100	0.225	0.192	0.200	0.202
13	14	13	92	8192	3,100	0.219	0.188	0.194	0.196

In figure 6, we present results for hypercubes. Since the local preference decreases as the dimension increases, the large deviation bound is tightened as the dimension increases. The embedded tree is then very close to a balanced tree with degree l and the asymptotic efficiency is expected to be close to the large deviation bound as l goes to infinity. The worst case bound is an accurate approximation when the dimension is small and is even better than the η_2 approximation. The errors of the η_2 approximation are within 10 percent.

6 Conclusions and Future Research

In this paper we have derived a lower bound for the efficiency of executing a task graph that corresponds to an iterative algorithm executing in parallel on multiple processors and requiring only partial synchronization. Many scientific computations can be modeled in this manner. Our results show, under certain probabilistic assumptions, that the asymptotic efficiency of such a computation is non-zero. This result contrasts with the case of full or barrier synchronization schemes where we showed the asymptotic efficiency is zero if task service times are unbounded. A fundamental property of the efficiency of partial synchronization is that it decreases as the computation graph becomes more spread out. This notion can be precisely defined for some graphs by the theory of majorization. The number of nodes reachable in two iterations is a measure of the amount a computation graph is spread out and can be used as a simple rule of thumb to estimate efficiency.

One of our contributions is to provide a mathematical approach for partial synchronization. The difficulty of the analysis lies in the fact that the times processors start their i'th iteration differ which is not the case for barrier synchronization. This makes an exact analysis intractable and we define a supermartingale that we use to obtain a lower bound on efficiency.

A question posed by this research concerns how one determines a task graph for a particular algorithm on a particular architecture. This mapping problem is, in general, very difficult and heuristics are often used. A consequence of our results shows that one factor to consider in such heuristics is how spread-out the computation graph is.

Acknowledgements The authors would like to thank Luigi Brochard and Frederica Darema for bringing the problem of partial synchronization to our attention and useful discussions. Ramesh Natarajan and Vijay Naik were very helpful in discussing applications of this work. We also gratefully acknowledge the detailed and insightful comments made by Steve Lavenberg after a careful reading of the manuscript. Lastly, we thank the referees for their careful and insightful comments which enhanced the presentation of the work.

A Appendix A

In this appendix we prove Theorem 4.2. First, we generalize the notion of Bernoulli loops and allow the probability vector $\underline{p}$ to be different at each iteration. In words, instead of using the same probability vector $\underline{p}$ for all iterations, we use the probability vector $\underline{p}_n$ to generate the arcs, $L_n(i,j)$, between the n-th and $n+1$-st iterations. Thus, we can characterize a Bernoulli loop by the parameters, $p_{n,k}$, $n = 1, \ldots, N-1$, $k = 1, \ldots, K$, and r. Note that loops constructed in this way are not necessarily i.i.d. loops since $\{L_n(i,j)\}$ are not i.i.d. random vectors. However, this generalization allows us to compare two loops which differ from each other only in the parameters for the n-th iteration, i.e., $\underline{p}_n$.

Now consider a Bernoulli loop generated by the parameters $\underline{p}_n$, $n = 1, \ldots, N-1$, and r. If we relabel the node $(n+1, 1)$ by $(n+1, 2)$ and the node $(n+1, 2)$ by $(n+1, 1)$, then we obtain another Bernoulli loop with parameter $\underline{p}_n$ replaced by $(p_{n,2}, p_{n,1}, p_{n,3}, \ldots, p_{n,K})$. Thus, the Bernoulli loops have the following property which we will use in the proof of Theorem 4.2.

Lemma A.1 *Let G_1 (G_2) be a Bernoulli loop with parameters $p^1_{n,j}$ ($p^2_{n,j}$) and r. If $(p^2_{n,1}, \ldots, p^2_{n,K})$ is a permutation of $(p^1_{n,1}, \ldots, p^1_{n,K})$ and the service times $X_{n,k}$ are i.i.d., then the completion times of these two loops are identically distributed.*

Proof of Theorem 4.2: First, we note that if $\underline{p}^1 \prec \underline{p}^2$, $\underline{p}^1$ can be derived from $\underline{p}^2$ by applications of a finite number of T-transformations (see Marshall and Olkin [29], p. 107), that is, transformations T having for some $1 \le i < j \le K$ and some $\alpha \in [0, 1]$, the form

$$T(\underline{p}^2) = (p^2_1, \ldots, p^2_{i-1}, \alpha p^2_i + (1-\alpha) p^2_j, p^2_{i+1}, \ldots, p^2_{j-1}, (1-\alpha) p^2_i + \alpha p^2_j, p^2_{j+1}, \ldots, p^2_K).$$

Thus, it suffices to consider the case $p^1_j = p^2_j$, $j = 3, \ldots, K$, $p^1_1 + p^1_2 = p^2_1 + p^2_2$, $p^2_1 \ge \max\{p^1_1, p^1_2\}$.

Now consider a sequence of Bernoulli loops $\{G^{(m)}, m = 1, \ldots, N\}$, with parameters $p^{(m)}_{n,j} = p^2_j$, $n = 1, \ldots, m-1$ and $p^{(m)}_{n,j} = p^1_j$, $n = m, \ldots, N-1$. Let $T^{(m)}_n$ be the completion time of the n-th iteration at $G^{(m)}$. Clearly, $T^{(1)}_N =_{st} T^1(N, K)$ and $T^{(N)}_N =_{st} T^2(N, K)$ ($=_{st}$ denotes that they have the same distribution). Thus, it suffices to show that $Ef(T^{(m)}_n) \ge Ef(T^{(m+1)}_n)$ for $n = 1, \ldots, N$ and f increasing.

Let $B^{(m)}_{n,k}$ be the completion time of the n-th iteration on processor k at $G^{(m)}$, $m = 1$ and 2. Thus, $T^{(m)}_n = \max_k \{B^{(m)}_{n,k}\}$. Now we would like to construct four Bernoulli loops, $\hat{G}^s, s = 1, 2, 3$ and 4, on the same probability space such that the completion time of the n-th iteration at $\hat{G}^s$, $\hat{T}^s_n$, satisfies

(T1) $\hat{T}^1_n =_{st} \hat{T}^2_n =_{st} T^{(m)}_n$,

(T2) $\hat{T}^3_n =_{st} \hat{T}^4_n =_{st} T^{(m+1)}_n$,

(T3) either $\hat{T}^1_n \ge \hat{T}^3_n$, $\hat{T}^2_n \ge \hat{T}^4_n$ for all n, or $\hat{T}^1_n \ge \hat{T}^4_n$, $\hat{T}^2_n \ge \hat{T}^3_n$ for all n.

Clearly, under the construction, we have either $f(\hat{T}_n^1) \geq f(\hat{T}_n^3)$, $f(\hat{T}_n^2) \geq f(\hat{T}_n^4)$, or $f(\hat{T}_n^1) \geq f(\hat{T}_n^4)$, $f(\hat{T}_n^2) \geq f(\hat{T}_n^3)$, for all f increasing. Taking expectations on the inequalities and using (T1) and (T2) yields $Ef(T_n^{(m)}) \geq Ef(T_n^{(m+1)})$ for all f increasing.

For simplicity, we show the construction for the case $m = 1$ first. The other cases follow similarly. The construction used in our proof can most easily be followed from Figure A.1.

Let $q_j \stackrel{\text{def}}{=} \sum_{l=1}^{j} p_l^1 = \sum_{l=1}^{j} p_l^2$, $j = 2, \ldots, K$. Consider the following sets, $\hat{A}_j^s$, $s = 1, 2, 3$ and 4 and $j = 1, \ldots, K$ (see figure A.1).

(A1) $\hat{A}_j^s = [q_{j-1}, q_j)$, $j = 3, \ldots, K$, for all $s = 1, 2, 3$ and 4.

(A2) Partition $[0, q_2)$ into four regions, R_1, R_2, R_3 and R_4, such that $R_1 = [0, p_1^2 - p_1^1)$, $R_2 = [p_1^2 - p_1^1, p_2^1)$, $R_3 = [p_2^1, p_1^2)$ and $R_4 = [p_1^2, q_2)$.

$$\hat{A}_1^1 = R_2 \bigcup R_3,\ \hat{A}_2^1 = R_1 \bigcup R_4,$$
$$\hat{A}_1^2 = R_1 \bigcup R_2,\ \hat{A}_2^2 = R_3 \bigcup R_4,$$
$$\hat{A}_1^3 = R_1 \bigcup R_2 \bigcup R_3,\ \hat{A}_2^3 = R_4,$$
$$\hat{A}_1^4 = R_2,\ \hat{A}_2^4 = R_1 \bigcup R_3 \bigcup R_4,$$

Construct independent random variables $\{U_{i,l}, i = 1, \ldots, K, l = 1, \ldots, r\}$ uniformly distributed over [0,1]. Let $\hat{L}_1^s(i,j) = 1$ if $U_{i,l} \in \hat{A}_j^s$ for some l. Note that $\hat{L}_1^s(i,j)$, $s = 1, 2, 3$ and 4 are constructed with the probability vectors $\underline{p}^1$, $(p_2^1, p_1^1, p_3^1, \ldots, p_K^1)$, $\underline{p}^2$ and $(p_2^2, p_1^2, p_3^2, \ldots, p_K^2)$, respectively. The task service times at the first iteration are constructed to be the same, i.e., $\hat{X}_{1,k}^s = X_{1,k}$ for all s and k. Also, the task service times at the second iteration are constructed to be the same for $k \geq 3$, i.e., $\hat{X}_{2,k}^s = X_{2,k}$ for all s and $k \geq 3$. It then follows from (3) that $\hat{B}_{2,k}^s$, $s = 1, 2, 3$ and 4 (the constructed completion times) are the same for $k \geq 3$ since $\hat{L}_1^s(i,k)$, $s = 1, 2, 3$ and 4 are the same for $k \geq 3$. Now we would like to show that under our construction either (i) $(\hat{B}_{2,1}^1, \hat{B}_{2,2}^1) \geq (\hat{B}_{2,1}^3, \hat{B}_{2,2}^3)$ and $(\hat{B}_{2,1}^2, \hat{B}_{2,2}^2) \geq (\hat{B}_{2,1}^4, \hat{B}_{2,2}^4)$ or (ii) $(\hat{B}_{2,1}^1, \hat{B}_{2,2}^1) \geq (\hat{B}_{2,1}^4, \hat{B}_{2,2}^4)$ and $(\hat{B}_{2,1}^2, \hat{B}_{2,2}^2) \geq (\hat{B}_{2,1}^3, \hat{B}_{2,2}^3)$.

Let $Y_k^s = \max_{\{i:\hat{L}_1^s(i,k)=1\}}\{X_{1,i}\}$. From (3), it follows that $\hat{B}_{2,k}^s = \hat{X}_{2,k}^s + Y_k^s$. Now consider all the uniformly distributed random variables, $U_{i,l}$, that fall in the interval $[0, q_2]$. Let U_{i^*,l^*} (for some i^* and l^*) be one of such random variables with the longest service time at the first iteration, i.e., $X_{1,i^*} \geq X_{1,i}$ for all i and l, $U_{i,l} \in [0, q_2]$. If there is a tie, then arbitrarily choose one of them. There are four cases needed to be considered: $U_{i^*,l^*} \in R_j$, $j = 1, 2, 3$ and 4. Here we only show the cases $j = 1$ and 2. The proof for the case $j = 3$ (resp. $j = 4$) is similar to that for $j = 1$ (resp. $j = 2$).

To motivate our method of proof, we informally discuss one case with the aid of Figure A.1. Suppose there is an arc from processor $P^\star$ that has the longest service time at the first iteration into region R_1 of the figure. By construction, this corresponds to an arc from $i^\star$ into processor 1 for systems 2 and 3 and an arc into processor 2 for systems 1 and 4. We set the service times for these respective processors in the second iteration to be equal and thus they finish their second iterations at the same time. We also set the service times for processor 2 in systems 2 and 3, and processor 1 for systems 1 and 4 to be equal in the second iteration. We claim that this construction implies that processor 1 finishes no later in system 4 than in system 1. To see this, note that arcs from processors of iteration 1 can delay processor 1 in system 1 or 4 by landing in region 2 but, additionally, for system 1 an arc falling in region 3 can hold up processor 1. Thus there is a greater chance that processor 1 is held up for system 1 than for system 4. A similar argument shows that processor 2 finishes no later in system 3 than in system 2. The proof given below, is a rigorous statement of these observations.

(i) $U_{i^*,l^*} \in R_1$. In this case, $Y_2^1 = Y_2^4 = Y_1^2 = Y_1^3 = X_{1,i^*}$. Since $\hat{A}_2^2 \supset \hat{A}_2^3$ and $\hat{A}_1^1 \supset \hat{A}_1^4$, $Y_2^2 \geq Y_2^3$ and $Y_1^1 \geq Y_1^4$. Now, we construct $\hat{X}_{2,2}^1 = \hat{X}_{2,2}^4 = \hat{X}_{2,1}^2 = \hat{X}_{2,1}^3 = X_{2,1}$ and $\hat{X}_{2,1}^1 = \hat{X}_{2,1}^4 = \hat{X}_{2,2}^2 = \hat{X}_{2,2}^3 = X_{2,2}$. Clearly, we have that $\hat{B}_{2,2}^1 = \hat{B}_{2,2}^4 = \hat{B}_{2,1}^2 = \hat{B}_{2,1}^3 = X_{1,i^*} + X_{2,1}$ and that $\hat{B}_{2,2}^2 \geq \hat{B}_{2,2}^3$ and $\hat{B}_{2,1}^1 \geq \hat{B}_{2,1}^4$.

(ii) $U_{i^*,l^*} \in R_2$. In this case, $Y_1^1 = Y_1^2 = Y_1^3 = Y_1^4 = X_{1,i^*}$. Moreover, note that $\max\{Y_2^1, Y_2^2\} = Y_2^4$ since $\hat{A}_2^1 \cup \hat{A}_2^2 = \hat{A}_2^4$ and that $\min\{Y_2^1, Y_2^2\} \geq Y_2^3$ since $\hat{A}_2^1 \supset \hat{A}_2^3$ and $\hat{A}_2^2 \supset \hat{A}_2^3$. Now we construct $\hat{X}_{2,1}^1 = \hat{X}_{2,1}^2 = \hat{X}_{2,1}^3 = \hat{X}_{2,1}^4 = X_{2,1}$ and $\hat{X}_{2,2}^1 = \hat{X}_{2,2}^2 = \hat{X}_{2,2}^3 = \hat{X}_{2,2}^4 = X_{2,2}$. Clearly, we have that $\hat{B}_{2,1}^1 = \hat{B}_{2,1}^4 = \hat{B}_{2,1}^2 = \hat{B}_{2,1}^3 = X_{1,i^*} + X_{2,1}$ and that $\max\{\hat{B}_{2,2}^1, \hat{B}_{2,2}^2\} = \hat{B}_{2,2}^4$ and $\min\{\hat{B}_{2,2}^1, \hat{B}_{2,2}^2\} \geq \hat{B}_{2,2}^3$.

From the second iteration onwards, we use the same construction of the arcs and task service times for these four task graphs, $\hat{G}^s$, $s = 1, 2, 3$ and 4. In words, $\hat{L}_n^s(i,j)$ and $\hat{X}_{n+1,i}^s$, $s = 1, 2, 3$ and 4, are the same for $n \geq 2$. From (3), it follows that for all n either (i) $(\hat{B}_{n,1}^1, \ldots, \hat{B}_{n,K}^1) \geq (\hat{B}_{n,1}^3, \ldots, \hat{B}_{n,K}^3)$ and $(\hat{B}_{n,1}^2, \ldots, \hat{B}_{n,K}^2) \geq (\hat{B}_{n,1}^4, \ldots, \hat{B}_{n,K}^4)$ or (ii) $(\hat{B}_{n,1}^1, \ldots, \hat{B}_{n,K}^1) \geq (\hat{B}_{n,1}^4, \ldots, \hat{B}_{n,K}^4)$ and $(\hat{B}_{n,1}^2, \ldots, \hat{B}_{n,K}^2) \geq (\hat{B}_{n,1}^3, \ldots, \hat{B}_{n,K}^3)$. Since $\hat{T}_n^s = \max_k\{\hat{B}_{n,k}^s\}$, (T3) is concluded. That (T1) and (T2) are satisfied is due to Lemma A.1. Thus, we complete the proof for $m = 1$. For $m \neq 1$, we observe that the only difference between $G^{(m)}$ and $G^{(m+1)}$ is the m-th iteration.

Therefore, we can construct the four task graphs to be the same up to the $(m-1)$-th iteration and then follow the construction for $m=1$. ■

B Appendix B

In this appendix we prove Theorem 4.5 using a result for "balanced trees" (defined below). The basic idea is to show that under the condition in Theorem 4.5 there exist an embedded "balanced tree" that achieve the lower bound. In the following, we introduce the notion of "balanced i.i.d. trees".

Definition B.1 *A random task graph with a single vertex (task) is called a balanced tree with level* 1.

A random task graph, G^{tr}, *is called a balanced tree with level* N *and degree* r *if*

(i) *there is a vertex called the root of the tree,*

(ii) *the remaining vertices are partitioned into* r *balanced trees with level* $N-1$ *and degree* r,

(iii) *there is an arc (precedence constraint) from the root to the root of each tree with level* $N-1$ *and degree* r, *(without loss of generality, we label the tree in the way that there is an arc from* (n,k) *to* $(n+1,(k-1)r^{n-1}+l)$ *for* $l=1,\dots,r$*)*

(iv) *the service time of a task* (n,k), $n=1,\dots,N$, $k=1,\dots,r^{n-1}$, *is denoted by the nonnegative random variable* $X_{n,k}$.

Definition B.2 *A balanced tree with level* N *and degree* r *is i.i.d. if* $\{X_{n,k}\}$ *are i.i.d.*

Let $B^{tr}_{n,k}$ be the sum of the task service times from the root (1,1) to the vertex (n,k). Let $T^{tr}_n \stackrel{\text{def}}{=} \max_k \{B^{tr}_{n,k}\}$. Thus, T^{tr}_n is the completion time of the n-th level.

Theorem B.3

$$\lim_{n\to\infty} \frac{T^{tr}_n}{n} = \frac{1}{\beta_r}.$$

The proof of Theorem B.3 can be found in [13] using a modified approach from [27] or in [9].

Now we proceed with the proof for Theorem 4.5. Consider the induced graph of a Bernoulli i.i.d. loop by the descendants of $(1,1)$ up to n levels. In words, every node (n,k) in the induced graph is connected by at least one path of the form,

$$(1,1) \mapsto (2,k_2) \mapsto \ldots \mapsto (n-1,k_{n-1}) \mapsto (n,k).$$

Clearly, the induced graph is a subgraph of the original loop. Thus, the completion time of the original loop is not less than the "longest" path in the induced graph, where the path length in the induced graph is meant to be the sum of the task service times along the path from (1,1) to (n,k). Observe that the induced graph is not a tree if more than one edge points to the same processor. Thus, the probability that the induced graph is not a balanced tree with level n and degree r is not greater than $\sum_{j=1}^{K}(p_j^{(K)})^2$ if the total number of arcs in the induced graph, $(r^n-1)/(r-1)$, is not greater than the total number of processors, K. Observe that

$$\sum_{j=1}^{K}(p_j^{(K)})^2 \leq \max_j p_j^{(K)} \sum_{j=1}^{K} p_j^{(K)} = \max_j p_j^{(K)}.$$

Since we assume that $\lim_{K\to\infty} p_j^{(K)} = 0$ for all j, the probability that the induced graph is a balanced tree with level n and degree r approaches 1 as K goes to infinity. Theorem 4.5 then follows from Theorem B.3. ■

References

[1] G.M. Amdahl, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities", *AFIPS Conf. Proc.*, Vol. 30, Thompson, Washington D.C., pp. 483-485, 1967.

[2] M. Avriel, "Nonlinear Programming: Analysis and Methods", *Prentice-Hall*, 1976.

[3] T. Axelrod, "Effects of Synchronization Barriers on Multiprocessor Performance", *Parallel Computing*, 3,pp.129-140, 1986.

[4] F. Baccelli and A.M. Makowski, "Queueing Models for Systems with Synchronization Constraints", *Proceedings of the IEEE*, 77(1),pp.138-161, 1989.

[5] F. Baccelli and Z. Liu, "On the Execution of Parallel Programs on Multiprocessor Systems-A Queueing Theory Approach," *J.A.C.M.*, Vol. 37, pp. 373-414, 1990.

[6] F. Baccelli, G. Cohen, G. Oslder, and J. Quadrat, "Synchronization and Linearity an Algebra for Discrete Event Systems," Wiley, 1992.

[7] F. Baccelli and Z. Liu, personal communication.

[8] S.H. Bokhari, "On the Mapping Problem", *IEEE Trans. on Computers*, Vol. C-30, No. 3,pp.207-214, March 1981.

[9] J.D. Biggins, "he First and Last Birth Problem for a Multitype Age-Dependent Branching Processes," *Adv. Appl. Prob.*, Vol. 8, pp. 446-459, 1976.

[10] L. Brochard, "Scalability, Granularity and Parallelism of Numerical Algorithms", *IBM Research Report RC 14786*, June 89.

[11] L. Brochard, "Efficiency of Some Parallel Numerical Algorithms on Distributed Systems", *Parallel Computing*, 12,pp.21-44, 1989.

[12] L. Brochard, J.P. Prost, and F. Faurie, "Synchronization and Load Unbalance Effects of Parallel Iterative Algorithms", *1989 Intern. Conf. on Parallel Processing*, pp.III-153-III-160.

[13] C.-S. Chang and R.D. Nelson, "Bounds on the Speedup and Efficiency of Partial Synchronization in Parallel Processing Systems," *IBM RC 16474. Jan. 1991.*

[14] R. Cytron, "Useful Parallelism in a Multiprocessing Environment", *Proc. 1985 Parallel Processing Conf.*.

[15] G. Dahlquist and A. Bjorck, "Numerical Methods", *Prentice-Hall*, 1974.

[16] H.A. David, "Order Statistics",*Wiley*, 1980.

[17] M. Dubois and F. Briggs, "Performance of Synchronized Iterative Processes in Multiprocessor Systems," *IEEE Trans. on Software Engineering*, Vol. SE-8, No. 4, pp.419-431, July 1982.

[18] M. Dubois, C. Scheurich and F. Briggs, "Synchronization, Coherence, and Event Ordering in Multiprocessors", *IEEE Computer*, 21,pp.9-21, February 1988.

[19] D. L. Eager, J. Zahorjan and E.D. Lazowska, "Speedup versus Efficiency in Parallel Systems", *IEEE Trans. on Computers*, 1989

[20] H Flatt, "A Simple Model for Parallel Processing", *IEEE Computer*, 17,pp.95, November 1984.

[21] H Flatt and K. Kennedy, "Performance of Parallel Processors", *Parallel Computing*, 12,pp.1-20, 1989.

[22] D.B. Gannon and J.Van Rosendale, "On the Impact of Communication Complexity on the Design of Parallel Numerical Algorithms", *IEEE Trans. on Computers*, VOl. C-33, No. 12, pp.1180-1194, December 1984.

[23] A. Greenbaum, "Synchronization Costs on Multiprocessors", *Parallel Computing*, 10,pp.3-14, 1989.

[24] J.L. Gustafson, "Reevaluating Amdahl's Law", *Comm. ACM.*, pp. 532-533, May 1988.

[25] J. Hack, "On the Promise of General Purpose Parallel Computing",*Parallel Computing*, 10, pp.261-275, 1989.

[26] S.S. Lavenberg, "Computer Performance Modeling Handbook", *Academic Press*, 1983.

[27] J.F.C. Kingman, "The First Birth Problem for an Age-Dependent Branching Process", *The Annals of Probability*, Vol 3, No 5, pp. 790-801, 1975.

[28] B. Lint and T. Agerwala, "Communication Issues in the Design and Analysis of Parallel Algorithms", *IEEE Trans. on Software Eng.*, Vol. SE-7, No 2, pp.174-181, March 1981.

[29] A.W. Marshall and I. Olkin , "Inequalities: Theory of Majorization and Its Applications", *Academic Press*, 1979.

[30] V. Naik, IBM T. J. Waston Research Center, personal communication.

[31] R. Nelson, D. Towsley, A.N. Tantawi, "Performance Analysis of Parallel Processing Systems," *IEEE Transactions on Software Engineering*, **14**, 4, 532-540 (April 1988).

[32] R. Nelson, A.N. Tantawi, "Approximate analysis of Fork/Join synchronization in parallel queues", *IEEE Trans. Computers*, C-37, pp. 739-743, 1988.

[33] R. Nelson, "A Performance of a General Parallel Processing Model", *Performance Evaluation Review*, Vol 18, no 1, pp.13-26, 1990.

[34] G.F. Pfister and V.A. Norton, "Hot Spot Contention and Combining in Multistage Interconnection Networks", *IEEE Trans. on Computers*, Vol. C-34, No 10, pp.943-948, Oct. 1985.

[35] J.M. Ortega and R. Voight, "Solution of Partial Differential Equations on Vector and Parallel Computers", *SIAM*, 1985.

[36] J. Robinson, "Some Analysis Techniques for Asynchronous Multiprocessor Algorithms", *IEEE Transactions on Software Engineering*, **Se-5**, 1, 24-31, (Jan. 1979).

[37] S. Ross, "Stochastic Processes", *Wiley*, 1983.

[38] H. Stone, “High Performance Computer Architectures”, *Addison-Wesley*, 1987.

[39] D. Stoyan, “Comparison Methods for Queues and other Stochastic Models”, *Wiley*, 1983.

[40] S.R.S. Varadhan, “Large Deviations and Applications”, *Society for Industrial and Applied Mathematics*, 1984.