

Modeling Communication Locality in Multiprocessors¹

C. Salisbury, Z. Chen, and R. Melhem

*Department of Computer Science, University of Pittsburgh,
Pittsburgh, Pennsylvania 15260*

E-mail: salisbur@cs.pitt.edu; zchen@transarc.com; melhem@cs.pitt.edu

Received November 13, 1997; revised October 26, 1998; accepted October 27, 1998

Locality of reference is an important aspect of many computer operations. It is often exploited to optimize the performance of computer functions. In this paper, we apply the locality concept to the communication patterns of parallel programs operating over an interconnection network with a fixed communication latency between any pair of attached nodes. Unbuffered multistage networks and all-optical networks are examples of these. We quantify the notions of spatial and temporal locality in this context, and combine them in a locality measure. This measure is used as the basis for identifying the communication working sets of a parallel program. We focus on programs with a looping structure and investigate conditions under which each working set consists of the complete set of paths required by a single loop. © 1999 Academic Press

Key Words: locality of reference; working sets; compiled communication; communication cost; interconnection networks; multiprocessor communication.

1. INTRODUCTION

Locality of reference was first identified as being important in the context of managing a program's use of pages of physical memory [6]. It has since been recognized as a concept that can be applied to a wide variety of contexts and has been exploited to improve the performance of many functions of modern computers. It is the fundamental reason for the effectiveness of caches used for both memory and file accesses. It has been used to improve performance of communication networks by caching the routing associated with network addresses. Locality of reference also exists in the patterns of communication over a multiprocessor interconnection network.

The effectiveness of various strategies to exploit memory access locality are often measured in terms of hit ratios for caches of various designs. This treatment of locality produces a result that is architecture and machine dependent. In

¹ This work is supported in part by NSF Award MIP-9633729 to the University of Pittsburgh.

some studies, the locality of a sequence of memory references has been evaluated independent from cache design, as in [8, 15, 21]. A key characteristic of memory accesses is that sequential addresses are likely to be accessed in order.

In the realm of communications, there may not be a significant relationship between processors located at sequential network addresses, in terms of either physical location or access patterns. Thus the spatial aspects of communication locality differ from those of memory accesses. On a local area network (LAN), message traffic is driven by end-user activity and can be highly variable. Traffic frequently reflects a client-server relationship between devices on the network. Quantitative measures of locality in communication over a local area network have been described in [9].

Communication patterns on the interconnection network of a parallel processing system have additional characteristics. While LAN traffic is driven by end-user activity, communication in an interconnection network is the result of the behavior of a single program constructed by a compiler. Interprocessor communication is required to coordinate tasks executing on different processors. The exploitation of locality through assignment of tasks to processors and the impact of data distribution on communication for cache coherence have been examined in numerous studies, including [7, 18, 20]. Communication in parallel programs often involves simultaneous, synchronized communication between processors and repetitive communication patterns arising from looping programs. Different loops may incorporate common algorithms for processor coordination and hence have similar communication patterns. While communication patterns that result in contention for network resources (i.e., "hot spots") can also be considered a form of locality, our focus is on network-independent characteristics arising from program behavior.

Communication locality in multiprocessor interconnection networks such as a k -ary n -cube is often described by the number of links between communicating processors. The performance impact of physical distance in an interconnection network with nonuniform communication latency was studied in [11]. Communication locality is frequently handled in more qualitative terms. Communication that is equally likely between any source and any destination has little locality. Communication between nearest neighbors along one or more dimensions has high locality. This is reflected not only in the reduced number of communicating pairs, but in the reduction in average number of links traversed by each message. Multiprocessor performance models have been developed to accommodate these kinds of assumptions about communication locality [1, 10]. Other researchers have recognized the importance of locality by designing multihop networks that reduce cost by providing fewer communication links between processors that are unlikely to communicate [4, 17].

We are interested in networks where the delay between each pair of processors is fixed. Such networks include unbuffered multistage networks and all-optical networks such as passive stars and WDM networks. Communication between any pair of processors has the same latency in these networks. Moreover, in many recent systems such as the Cray T3D physical distance may not be a significant factor in network performance [2, 16]. This requires a different approach to

describing locality than what is used in the multihop networks. To be applicable to a variety of network architectures, we seek to understand locality and communication working sets independent of the network architecture. We will quantify some intuitive notions of locality and see how broadly the results can be applied.

In some cases, techniques called *compiled communication* can be performed on parallel programs to identify static communication patterns prior to program execution [3, 12, 14, 22]. For this reason, we will consider the locality of a predetermined sequence of communication requests that will be presented to an interconnection network. In particular, we define a quantitative measure that satisfies some intuitive properties of locality and can be applied to parallel processor communication patterns over networks with uniform latency. The measure provides a way to compare sequences and determine which has more locality.

More important than locality, however, is a program's communication performance. Although the locality measure is network independent, to be useful it should be capable of predicting performance. Networks that use multiplexing techniques have performance characteristics that depend on the number of connections provided simultaneously. Using a model of compiled communication performance, we show there is a strong relationship between locality and performance on a multiplexed network.

Techniques for compiled communication could therefore incorporate a locality analysis to assist in identifying communication working sets and optimize performance. On distributed shared memory systems, such analysis could assist with development of a data distribution that minimizes communication delays [13]. Similarly, when communication patterns are established by the programmer, a locality analysis could be used to assist in the selection of an algorithm with good communication performance.

In Section 2 we present a model of the communication pattern of a parallel application. In Section 3, we quantify the concept of communication locality of reference and show how to identify the working set of communication paths used by an application. We develop the results more fully for the special case of looping programs in Section 4, where we show how locality is affected by the number of loop iterations and by the reoccurrence of paths in different loops. We show that our measure of locality is related to communication performance in a circuit-switched, time-division-multiplexed network in Section 5. Our conclusions are in Section 6.

2. THE APPLICATION MODEL

To the interconnection network, an application appears as a sequence of requests for a network connection. These requests can originate on any node attached to the network. Each request can be characterized by its source, destination, message length, and arrival time. We model the requests using fixed length messages and represent arrival times by ordering messages in the sequence in which they are presented to the network. The network processes each request by providing a path connecting the message source and destination. We will represent the sequence of communication connection requests by $\mathcal{T} = p_1, p_2, \dots$, where p_i represents the path connecting the source/destination pair (s_i, d_i) for the i th communication request.

To express the looping structure common to parallel applications, we let T_i be the sequence of requests generated by one iteration of loop i and r_i be the number of iterations. We denote the sequence of requests generated by this loop as $T_i^{r_i}$, indicating that the sequence T_i is repeated r_i times. When $r_i=1$, the sequence T_i does not repeat and r_i can be omitted from the notation. The requests from an application with n loops can be described by

$$\mathcal{F} = T_1^{r_1} T_2^{r_2} \dots T_n^{r_n}.$$

We will often be interested in the number of different paths required for communication. We define M_i to be the set of different paths used in loop i and the number of different paths to be $l_i = |M_i|$. We will also find it useful to deal with looping structures that have disjoint loops or distinct paths, as defined below.

DEFINITION 1. We say that a looping structure has *disjoint* loops if each loop uses paths that are not used by any other loop. Specifically, if M_i contains all the paths in T_i and M_j contains all the paths in T_j , then $M_i \cap M_j = \emptyset$ for all $i \neq j$.

DEFINITION 2. We say that a loop has *distinct* paths when each path in the loop is used exactly once in each iteration of the loop. Loops with distinct paths have $l_i = |T_i|$.

3. MODELING LOCALITY OF REFERENCE

To be useful for understanding communication performance, we would like to define a locality measure that encapsulates application characteristics related to the size of the network required to handle communication and the number of connection requests that can be satisfied before the network state must change. To be useful in a variety of contexts, the measure should depend only on program characteristics, and not on the characteristics of a network or how it is controlled. This suggests that the locality measure should be useful for determining a program's communication working set.

In order to identify the communication working sets of a program, we will make an estimate of the working sets and apply the locality measure. We do this by breaking a sequence of communication requests \mathcal{F} into n subsequences such that $\mathcal{F} = P_1, P_2, \dots, P_n$. Each subsequence P_i is called a *partition*, and the set of these partitions is a *partitioning* of \mathcal{F} . Each partition is an estimate of the working set at a different point in time. We denote a function which creates these partitions as $\mathcal{P}(\mathcal{F})$. A locality measure should characterize the network size and path reuse characteristics of the estimated working sets. A locality measure is therefore a function of both the sequence of requests and the manner in which the requests are partitioned. We define the locality of a partitioning and the locality of a sequence as follows.

DEFINITION 3. The *locality of a partitioning* refers to the value of the locality measure L on the partitioning of the sequence \mathcal{F} created by the function \mathcal{P} . This is denoted by $L(\mathcal{F}, \mathcal{P})$ and is called the *L-measure* of \mathcal{F} with \mathcal{P} .

DEFINITION 4. The *locality of a sequence* \mathcal{T} , denoted $\mathcal{L}(\mathcal{T})$, is equal to the greatest L -measure that can be obtained from a partitioning of \mathcal{T} ,

$$\mathcal{L}(\mathcal{T}) = \max_{\mathcal{P}} \{L(\mathcal{T}, \mathcal{P})\}.$$

DEFINITION 5. If \mathcal{P} is a partitioning function for which $\mathcal{L}(\mathcal{T}) = L(\mathcal{T}, \mathcal{P})$, then the partitioning $\mathcal{P}(\mathcal{T})$ is an *optimal* partitioning of \mathcal{T} .

The communication working sets of the parallel program are the partitions created by the optimal partitioning function. The objective of this study is to develop a general approach for locating the communication working sets of \mathcal{T} by determining an appropriate measure L and identifying an optimal partitioning function \mathcal{P} .

3.1. The Locality Measure

A communication pattern that is highly local is desirable for two reasons. First, high locality suggests that the communication working set has only a “small” number of different paths. Thus, the application does not use many network resources at any one time. This loosely corresponds to the notion of high spatial locality. Second, good locality also suggests that path utilization should be high, which means the paths provided should be reused often. This loosely corresponds to the notion of temporal locality. We combine these concepts of spatial and temporal locality in our locality measure. Small partition sizes coupled with high reuse of the paths in a partition means high locality. The choice of measure will necessarily reflect a balance between these two. That is, a reduction of spatial locality due to an increase in partition size can be offset by a gain in temporal locality due to an increase in the amount of reuse.

We are particularly interested in the locality of the looping programs commonly run on parallel systems. When a program has a single loop in which communication paths are distinct, the number of paths may be a reasonable measure of communication locality. However, this may not be an appropriate measure when paths are used multiple times within a loop, the program consists of a series of loops, or the communication is not from a looping program. We will therefore develop a more general measure that incorporates the notions of temporal and spatial locality of reference for an arbitrary communication pattern. In many cases, to gain insight into the characteristics of locality and to develop closed form solutions it will be necessary to consider communication patterns with a specific structure. When special communication patterns are used, the insight gained into the properties of locality should be applicable to other patterns as well.

Consider a sequence \mathcal{T} and a partitioning P_1, P_2, \dots, P_n . Let M_i be the set of all the different paths used in the partition P_i , and let $|M_i|$ and $|P_i|$ be the number of paths in M_i and P_i , respectively. We define spatial locality of a partitioning to be inversely proportional to the average number of different paths used in its partitions.

Fewer paths means greater spatial locality, reflecting a smaller demand for network paths. That is,

$$\text{Spatial Locality } (\mathcal{T}, \mathcal{P}) = \frac{1}{(1/n) \sum_{i=1}^n |M_i|}. \quad (1)$$

Temporal locality is expressed in terms of the reuse of paths within a partition. Path reuse for partition P_i is $|P_i|/|M_i|$. To encourage the creation of partitions with high path reuse as would be found in a working set, we use the root-sum-of-squares of the reuse to weight the temporal locality measure strongly toward such partitions. Thus, the temporal locality of a partitioning is defined as

$$\text{Temporal Locality } (\mathcal{T}, \mathcal{P}) = \frac{1}{n} \sqrt{\sum_{i=1}^n (|P_i|/|M_i|)^2}. \quad (2)$$

The locality of a partitioning is defined as the product of these two measures:

$$L(\mathcal{T}, \mathcal{P}) = \frac{\sqrt{\sum (|P_i|/|M_i|)^2}}{\sum |M_i|}. \quad (3)$$

While partitioning functions can create partitions arbitrarily, certain functions warrant special consideration. We define \mathcal{P}_S to be the function that places all communications into a single partition. We can also create n partitions so that, for some value m , $|M_i| = m$ for $i = 1, \dots, n-1$ and $|M_n| \leq m$. When the partitions are created sequentially so that P_i is the longest sequence for which $|M_i| = m$, we call this the m -path partitioning of \mathcal{T} and denote the partitioning function \mathcal{P}_{Mm} .

In the following examples we will use an integer to represent a single communication request on a particular path. For clarity, the examples will use short sequences \mathcal{T} and small values of m . It should be clear, however, that the concepts presented

TABLE 1
Partitioning Simple Sequences with Various Methods

Requests \mathcal{T}	Method	Partitions P_i	Path sets M_i	$L(\mathcal{T}, \mathcal{P})$
1, 2, 3, 4, 5, 6, 7, 8	\mathcal{P}_S	(1, 2, 3, 4, 5, 6, 7, 8)	{1, 2, 3, 4, 5, 6, 7, 8}	0.13
	\mathcal{P}_{M1}	(1), (2), (3), (4), (5), (6), (7), (8)	{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}	0.35
	\mathcal{P}_{M2}	(1, 2), (3, 4), (5, 6), (7, 8)	{1, 2}, {3, 4}, {5, 6}, {7, 8}	0.25
	Arbitrary	(1), (2, 3, 4), (5), (6), (7, 8)	{1}, {2, 3, 4}, {5}, {6}, {7, 8}	0.28
1, 2, 2, 3, 3, 3, 1, 4	\mathcal{P}_S	(1, 2, 2, 3, 3, 3, 1, 4)	{1, 2, 3, 4}	0.50
	\mathcal{P}_{M1}	(1), (2, 2), (3, 3, 3), (1), (4)	{1}, {2}, {3}, {1}, {4}	0.80
	\mathcal{P}_{M2}	(1, 2, 2), (3, 3, 3, 1), (4)	{1, 2}, {1, 3}, {4}	0.54
	Arbitrary	(1, 2, 2, 3), (3), (3, 1, 4)	{1, 2, 3}, {3}, {1, 3, 4}	0.28
1, 3, 2, 3, 2, 3, 1, 4	\mathcal{P}_S	(1, 3, 2, 3, 2, 3, 1, 4)	{1, 2, 3, 4}	0.50
	\mathcal{P}_{M1}	(1), (3), (2), (3), (2), (3), (1), (4)	{1}, {3}, {2}, {3}, {2}, {3}, {1}, {4}	0.35
	\mathcal{P}_{M2}	(1, 3), (2, 3, 2, 3), (1, 4)	{1, 3}, {2, 3}, {1, 4}	0.41
	Arbitrary	(1), (3, 2, 3, 2, 3), (1), (4)	{1}, {2, 3}, {1}, {4}	0.61

apply as well to arbitrarily large sequences and values of m . Table 1 shows the locality of simple path sequences with different partitionings.

EXAMPLE 1. A network that connects four inputs to four outputs has 16 possible paths through the network. These paths can be represented by the integers 1 through 16. The sequence $\mathcal{T} = 1, 2, 2, 1$ represents the use of path 1, followed by path 2, followed by the reuse of paths 2 and 1. An arbitrary partitioning of this sequence could be $P_1 = 1, 2, 2$ and $P_2 = 1$. In this case, $M_1 = \{1, 2\}$ and $M_2 = \{1\}$. Using $\mathcal{P}_S(\mathcal{T})$ we have $P_1 = 1, 2, 2, 1$ and $M_1 = \{1, 2\}$. With $\mathcal{P}_{M_1}(\mathcal{T})$ we obtain $P_1 = 1, P_2 = 2, 2, P_3 = 1$, and $M_1 = \{1\}, M_2 = \{2\}, M_3 = \{1\}$.

3.2. Partitioning Sequences

In some cases, we can describe how to modify a partitioning to increase locality. In Lemma 1, we show that locality will increase if we can divide a partition into two pieces which have no paths in common. Lemma 2 then describes when locality is increased by joining adjacent partitions.

LEMMA 1. *Let P_i be a partition of $\mathcal{P}(\mathcal{T})$. Let $\mathcal{P}'(\mathcal{T})$ be a refinement of $\mathcal{P}(\mathcal{T})$ formed by splitting partition P_i into P_{i1}, P_{i2} , so that $\mathcal{P}'(\mathcal{T}) = P_1, \dots, P_{i1}, P_{i2}, \dots, P_n$. Let M_{i1} and M_{i2} be the set of paths corresponding to P_{i1} and P_{i2} , respectively. If $M_{i1} \cap M_{i2} = \emptyset$ then $L(\mathcal{T}, \mathcal{P}') > L(\mathcal{T}, \mathcal{P})$.*

Proof. Since $M_{i1} \cap M_{i2} = \emptyset$ we know that $\sum_{\mathcal{P}'} |M_i| = \sum_{\mathcal{P}} |M_i|$. Thus the lemma will be true when

$$\left(\frac{|P_{i1}|}{|M_{i1}|}\right)^2 + \left(\frac{|P_{i2}|}{|M_{i2}|}\right)^2 > \left(\frac{|P_i|}{|M_i|}\right)^2.$$

Since $|P_i| = |P_{i1}| + |P_{i2}|$ and $|M_i| = |M_{i1}| + |M_{i2}|$, with some algebraic manipulation we can determine that this expression is always true. ■

LEMMA 2. *Let $\mathcal{P}(\mathcal{T})$ be a partitioning which has adjacent partitions P_i and P_{i+1} . Let M_i and M_{i+1} be the corresponding sets of paths. Let $\mathcal{P}'(\mathcal{T})$ be a partitioning identical to $\mathcal{P}(\mathcal{T})$ except that P_i and P_{i+1} are combined into a single partition. If $M_i \subseteq M_{i+1}$ and*

$$\frac{2|P_{i+1}|}{|P_i|} \geq \left(\frac{|M_{i+1}|}{|M_i|}\right)^2 - 1, \quad (4)$$

then $L(\mathcal{T}, \mathcal{P}') > L(\mathcal{T}, \mathcal{P})$.

Proof. The proof is straightforward, following the same approach used for Lemma 1. ■

When $M_i = M_{i+1}$ the condition of Lemma 2 is always met and we will always increase locality by combining adjacent partitions that use the same paths. When adjacent partitions use almost the same set of paths ($M_i \subset M_{i+1}$ and $|M_i| \approx |M_{i+1}|$), we should consider merging the partitions even when the sequence

lengths are very different ($|P_{i+1}| \ll |P_i|$). When the numbers of paths used by these partitions are very different ($|M_{i+1}| \gg |M_i|$) we should consider merging the partitions only when the partition with more paths has many more requests ($|P_{i+1}| \gg |P_i|$).

It is difficult to make general statements governing arbitrary sequences of requests. In the next section we focus on looping applications.

4. LOCALITY OF LOOPING SEQUENCES

In looping programs, it is natural to expect that the communication requests from each loop form a working set. Thus, we define the partitioning that places all requests from each loop into a separate partition to be the *natural partitioning* of the application and refer to it as \mathcal{P}_N . Note that $\mathcal{P}_N(\mathcal{T})$ is the partitioning that results from applying \mathcal{P}_S to each individual loop, T_i^r . Intuitively, this partitioning should have the greatest locality as long as the number of iterations of each loop is sufficiently large and the loops are sufficiently disjoint. To quantify these two conditions for general loop structures is rather complicated, if at all possible. Thus, in the next sections we will consider only special loop structures, beginning with loops that are disjoint.

4.1. Sequences of Disjoint Loops

We can show directly from the definition of the locality measure that \mathcal{P}_S is the optimal partitioning of a loop T^r with distinct paths when $r^3 > l$, where $l = |T|$. It is more difficult to determine when \mathcal{P}_S is optimal for other loops where $l < |T|$. However, we can place an upper bound on the minimum value of r needed to ensure that \mathcal{P}_S is the optimal partitioning.

THEOREM 3. *Let $\mathcal{T} = T^r$ be a sequence of communication requests, and let k be the maximum number of times any path is used in a single iteration of the loop. Then the single partitioning of the loop, $\mathcal{P}_S(\mathcal{T})$, is optimal when*

$$r \geq \frac{(kl)^2}{|T|}. \quad (5)$$

Proof. To show that the locality of $\mathcal{P}_S(\mathcal{T})$ is greater than the locality of an arbitrary partitioning $\mathcal{P}(\mathcal{T})$ with more than one partition, it is sufficient to show that both temporal locality and spatial locality are greatest with $\mathcal{P}_S(\mathcal{T})$. In taking this approach, we do not consider offsetting an increase in one locality with a decrease in the other.

For spatial locality, the value of $\sum_i |M_i|$ for $\mathcal{P}_S(\mathcal{T})$ is l . No other partitioning of the loop can have a smaller value, so that $\mathcal{P}_S(\mathcal{T})$ has the greatest spatial locality.

We can bound the amount of temporal locality attainable from a function that creates two or more partitions. Let P_i be the partitions created by $\mathcal{P}(\mathcal{T})$, and let M_i be the corresponding sets of paths. Place the partitions in two groups based on the size of M_i . Group 1 contains partitions where $|M_i| = l$ and group 2 contains partitions where $|M_i| < l$. Let q be the total number of communication requests in

group 2. Then $r|T| - q$ is the total number of communication requests in group 1. We can place an upper bound on the contribution from group 1 to the sum of squares in temporal locality, since

$$\sum_i \left(\frac{|P_i|}{|M_i|} \right)^2 = \sum_i \left(\frac{|P_i|}{l} \right)^2 \leq \left(\frac{\sum_i |P_i|}{l} \right)^2 = \left(\frac{r|T| - q}{l} \right)^2.$$

Similarly, we can bound the sum of squares contribution from partitions in group 2. There are at most q such partitions, and they have reuse no greater than k . For these partitions,

$$\sum \left(\frac{|P_i|}{|M_i|} \right)^2 \leq qk^2.$$

Combining these results for the two groups, we know that the temporal locality of $\mathcal{P}(\mathcal{T})$ is at most

$$qk^2 + \left(\frac{r|T| - q}{l} \right)^2.$$

The temporal locality of the single partitioning $\mathcal{P}_S(\mathcal{T})$ is $(r|T|/l)^2$. This exceeds the temporal locality of an arbitrary partitioning when $(kl)^2 + q \leq 2r|T|$. Since q cannot exceed the total number of requests $r|T|$, if $r \geq (kl)^2/|T|$ then $L(\mathcal{T}, \mathcal{P}_S) \geq L(\mathcal{T}, \mathcal{P})$ and $\mathcal{P}_S(\mathcal{T})$ is an optimal partitioning. ■

Note that for a loop with distinct paths where $l = |T|$ and $k = 1$, Theorem 3 provides the bound $r > l$, which is much higher than the $r^3 \geq l$ obtained directly from the locality definitions. This shows that requiring both spatial and temporal locality to be optimal produces a bound which is not tight. It is difficult to tighten this bound without placing additional restrictions on the sequence T .

We can, however, state when the natural partitioning of a sequence of disjoint loops is an optimal partitioning.

COROLLARY 3.1. *Let $\mathcal{T} = T_1^{r_1}, \dots, T_n^{r_n}$ be a sequence of communication requests from an application with n disjoint loops, and let k_i be the maximum number of times any path is used in a single iteration of loop i . Then the natural partitioning of the sequence, $\mathcal{P}_N(\mathcal{T})$, is optimal when*

$$r_i \geq \frac{(k_i l_i)^2}{|T_i|} \quad \text{for all } i.$$

For loops with distinct paths, $\mathcal{P}_N(\mathcal{T})$ will be optimal when $r_i \geq l_i$ for $i = 1, \dots, n$.

Proof. Since the paths required by the loops are disjoint, from Lemma 1 we know that the greatest locality is attained with partitions containing requests from only one loop. Since the number of iterations in each loop meets the requirement of Theorem 3, the natural partitioning of the sequence of loops is optimal. ■

TABLE 2
Partitions of Disjoint Looping Sequences

Sequence \mathcal{F}	Partitioning \mathcal{P}	$M_1, M_2, \dots, M_{\text{END}}$	END	$L(\mathcal{F}, \mathcal{P})$
$(1, 2, 3, 4)^{40}$	\mathcal{P}_S	$\{1, 2, 3, 4\}$	1	10.00
	\mathcal{P}_{M1}	$\{1\}, \{2\}, \{3\}, \{4\}, \{1\}, \dots$	160	0.08
$(1, 2, 3, 3)^{20} (4, 5, 6, 6)^{20}$	\mathcal{P}_N	$\{1, 2, 3\}, \{4, 5, 6\}$	2	6.28
	\mathcal{P}_S	$\{1, 2, 3, 4, 5, 6\}$	1	4.44
	\mathcal{P}_{M1}	$\{1\}, \{2\}, \{3\}, \{1\}, \dots, \{3\},$ $\{4\}, \{5\}, \{6\}, \{4\}, \dots, \{6\}$	120	0.13
	\mathcal{P}_{M2}	$\{1, 2\}, \{3, 1\}, \{2, 3\}, \dots$ $\{4, 5\}, \{6, 4\}, \{5, 6\}, \dots$	60	0.09
$(1, 2, 3, 4)^{30} (5, 6, 7, 8)^{10}$	\mathcal{P}_N	$\{1, 2, 3, 4\}, \{5, 6, 7, 8\}$	2	3.95
	\mathcal{P}_{M1}	$\{1\}, \{2\}, \{3\}, \{4\}, \{1\}, \dots, \{4\},$ $\{5\}, \{6\}, \{7\}, \{8\}, \{5\}, \dots, \{8\}$	160	0.08
$(1, 2, 3, 4)^{20} (5, 6, 7, 8)^{20}$	\mathcal{P}_N	$\{1, 2, 3, 4\}, \{5, 6, 7, 8\}$	2	3.54
	\mathcal{P}_S	$\{1, 2, 3, 4, 5, 6, 7, 8\}$	1	2.50
	\mathcal{P}_{M1}	$\{1\}, \{2\}, \{3\}, \{4\}, \{1\}, \dots, \{4\},$ $\{5\}, \{6\}, \{7\}, \{8\}, \{5\}, \dots, \{8\}$	160	0.08

Table 2 shows several looping sequences, the sets of paths required by various partitionings, and the resulting locality. For these sequences, the number of iterations meets the requirement of Theorem 3. The examples in Table 2 confirm that the natural partitioning has the greatest locality. The locality of the one and two path partitioning is very low because there is little or no path reuse, and temporal locality is low. The large number of partitions means that the sum of the partition sizes is large, making spatial locality low as well. Combining disjoint loops into a single partition also reduces locality compared to the natural partitioning.

4.2. Partitioning Sequences of Non-disjoint Loops with \mathcal{P}_N and \mathcal{P}_S

In this section we show when it is better to partition a sequence of loops with overlapping sets of paths by using \mathcal{P}_N rather than by Using \mathcal{P}_S . Lemma 2 can be applied when the paths used by one loop are a subset of the paths used by an adjacent loop. However, it cannot be applied when loops have some common paths as well as some different paths.

To evaluate partitions that combine a number of overlapping loops, we need additional information. The ratio of the number of communication requests in loop i to the total number of requests in the sequence, $r_i |T_i| / |\sum_{j=1}^n r_j |T_j|$, will be called u_i . When $u_i > u_j$, the locality characteristics of loop i will have a greater impact on the L -measure of a combined partitioning than will the locality characteristics of loop j . The total number of different paths in a sequence of k loops starting at loop j is equal to $|\cup_{i=j}^{j+k-1} M_i|$ and will be denoted λ_k . The value of j will be clear from the context. For example λ_n is the total number of different paths in the entire sequence of n loops, requiring $j=1$. The parameters used to characterize locality are summarized in Table 3.

TABLE 3
Locality Parameters

Parameter	Meaning
T_i	A sequence of communication requests in loop i
M_i	The set of different paths used in a sequence of requests
l_i	The number of different paths used in a sequence of requests. $l_i = M_i $
r_i	The number of iterations of a loop
u_i	The ratio of the number of communication requests in loop i to the total number of requests in the sequence, $r_i T_i / \sum_{j=1}^n r_j T_j $
λ_k	The total number of different paths in a sequence of k loops, $ \bigcup_{i=j}^{j+k-1} M_i $

The following theorem describes when a sequence of overlapping loops has a greater L -measure when the natural partitioning is used than when the loops are combined into a single partition.

THEOREM 4. *Let $\mathcal{F} = T_1^{r_1} T_2^{r_2} \dots T_n^{r_n}$. Define the decision function*

$$d(u_i, l_i, \lambda_n) \equiv \frac{\lambda_n^2}{\sum_1^n l_i} \sqrt{\sum_1^n u_i^2 / l_i^2}. \quad (6)$$

Then, $L(\mathcal{F}, \mathcal{P}_N) \geq L(\mathcal{F}, \mathcal{P}_S)$ when $d(u_i, l_i, \lambda_n) \geq 1$, and $L(\mathcal{F}, \mathcal{P}_S) \geq L(\mathcal{F}, \mathcal{P}_N)$ when $d(u_i, l_i, \lambda_n) \leq 1$. In particular, $L(\mathcal{F}, \mathcal{P}_N) \geq L(\mathcal{F}, \mathcal{P}_S)$ for any values of u_i whenever

$$\lambda_n^2 \geq \left(\sum_1^n l_i \right) \sqrt{\sum_1^n l_i^2}. \quad (7)$$

$L(\mathcal{F}, \mathcal{P}_S) \geq L(\mathcal{F}, \mathcal{P}_N)$ for any values of u_i whenever

$$\lambda_n^2 \leq \left(\sum_1^n l_i \right) \min_i \{l_i\}. \quad (8)$$

Proof. From the definitions, the locality of the natural partitioning is

$$L(\mathcal{F}, \mathcal{P}_N) = \frac{\sqrt{\sum_1^n ((r_i |T_i|) / l_i)^2}}{\sum_1^n l_i}$$

and that of the single partitioning is

$$L(\mathcal{F}, \mathcal{P}_S) = \frac{\sum_1^n r_i |T_i|}{\lambda_n^2}.$$

We construct the decision function $d(u_i, l_i, \lambda_n)$ as

$$d(u_i, l_i, \lambda_n) \equiv \frac{L(\mathcal{F}, \mathcal{P}_N)}{L(\mathcal{F}, \mathcal{P}_S)} = \frac{\lambda_n^2}{\sum_1^n l_i} \frac{\sqrt{\sum_1^n ((r_i |T_i|) / l_i)^2}}{\sum_1^n r_i |T_i|} = \frac{\lambda_n^2}{\sum_1^n l_i} \sqrt{\sum_1^n u_i^2 / l_i^2}.$$

When $d \geq 1$, $L(\mathcal{F}, \mathcal{P}_N) \geq L(\mathcal{F}, \mathcal{P}_S)$.

If we consider $d(u_i, l_i, \lambda_n)$ as a function of variables u_i with $\sum_1^n u_i = 1$, we can use the Lagrange multiplier to find the minimum value. Let

$$f(u_i) = d(u_i) + \gamma \left(\sum_1^n u_i - 1 \right);$$

then

$$f'_{u_i} = d'_{u_i} + \gamma = \frac{\lambda_n^2}{l_i^2 \sum_1^n l_i} \frac{u_i}{\sqrt{\sum_1^n u_i^2 / l_i^2}} + \gamma = 0. \quad (9)$$

Since $f'_\gamma = \sum_1^n u_i - 1 = 0$, the value of γ does not affect the location of the minimum. If we multiply Eq. (9) by l_i^2 / λ_n^2 and add the terms for all u_i we can determine the value of γ . Substituting this value back into Eq. (9), we find that the minimum value of $d(u_i)$ occurs when, for all i ,

$$u_i = \frac{l_i^2}{\sum_1^n l_i^2}. \quad (10)$$

Hence the minimum value of the decision function is $d_{\min} = \lambda_n^2 / (\sum_1^n l_i \sqrt{\sum_1^n l_i^2})$. The natural partitioning has a greater L -measure for any values of u_i whenever $d_{\min} \geq 1$. This occurs when

$$\lambda_n^2 \geq \left(\sum_1^n l_i \right) \sqrt{\sum_1^n l_i^2}.$$

Since d'_{u_i} is monotonic, the maximum value of $d(u_i)$ will be reached at a boundary point where $u_i = 1$ for some value of i . Hence $d_{\max} = (\lambda_n^2 / \sum_1^n l_i) \max_i \{1/l_i\}$. If $d_{\max} \leq 1$, then a single partition has the greater L -measure and for any values of u_i ,

$$\lambda_n^2 \leq \left(\sum_1^n l_i \right) \min_i \{l_i\}. \quad \blacksquare$$

This theorem says that when there is a lot of overlap between the loops, λ_n is small and a single partition has a greater L -measure. When there is little overlap, λ_n is large and the natural partitioning has a greater L -measure. In order to quantify this more, we must look at special cases. If $l_i = l$ for all i , we can use l/λ_n as a measure of overlap. This ratio is equal to $1/n$ when loops are disjoint and is equal to 1 when all the loops use the same paths. For this special case, we can simplify the result of Theorem 4.

COROLLARY 4.1. *Let $\mathcal{F} = T_1^r T_2^r \dots T_n^r$, where $l_i = l$ for all i . The natural partitioning has a greater L -measure than a single partition when $l/\lambda_n \leq 1/\sqrt{n}$. The single partition has a greater L -measure when $l/\lambda_n \geq 1/\sqrt{n}$.*

When l/λ_n exceeds a threshold value, the loops have many paths in common and a single partition is better regardless of the distribution of requests between loops.

When the ratio is below a different threshold, the loops are sufficiently different that the natural partitioning is better. For example if $n=4$ and $l=10$, the natural partitioning is best when $l/\lambda_n \leq 0.35$, which means that the total number of different paths in the loops is larger than 28. The single partitioning is best when $l/\lambda_n \geq 0.50$, corresponding to 20 or fewer different paths.

When there are only two loops, Theorem 4 identifies the optimal partitioning.

COROLLARY 4.2. *Let $\mathcal{T} = T_1^{r_1} T_2^{r_2}$ with $M_1 \cap M_2 \neq \emptyset$, where r_1 and r_2 satisfy the requirement of Theorem 3. The following decision function can be used to determine then the natural partitioning is optimal:*

$$d(u_1, u_2, \lambda_n, l_1, l_2) = \frac{\lambda_n^2}{l_1 + l_2} \sqrt{\left(\frac{u_1}{l_1}\right)^2 + \left(\frac{u_2}{l_2}\right)^2}. \quad (11)$$

If $d \geq 1$, $\mathcal{L}(\mathcal{T}) = L(\mathcal{T}, \mathcal{P}_N)$. Otherwise, $\mathcal{L}(\mathcal{T}) = L(\mathcal{T}, \mathcal{P}_S)$. In particular, \mathcal{P}_N is optimal when

$$\lambda_n^2 \geq (l_1 + l_2) \sqrt{l_1^2 + l_2^2}, \quad (12)$$

and \mathcal{P}_S is optimal when

$$\lambda_n^2 \leq (l_1 + l_2) \min\{l_1, l_2\}. \quad (13)$$

Figure 1 shows how the decision function is affected by the overlap and the distribution of requests between two loops. When both loops have the same number of different paths as shown in Fig. 1a, the minimum value of the decision function occurs when both loops have the same number of requests regardless of the amount of overlap. When the amount of overlap is small enough that $l/\lambda_n \leq 0.59$, the natural partitioning is optimal regardless of the distribution of requests. When $l/\lambda_n \geq 0.71$, the single partition is optimal. An alternative way to measure overlap is to use the ratio l_c/l , where l_c is the number of paths common to both loops. For two loops, $l_c = l_1 + l_2 - \lambda_n$. \mathcal{P}_N is optimal when $l_c/l \leq 0.32$ and the two loops have less than 32% of their paths in common. \mathcal{P}_S is optimal when $l_c/l \geq 0.59$ and the loops have more than 59% of their paths in common. For intermediate values of overlap, the decision function must be used to make the optimal choice based on the relative number of requests in the two loops.

Figure 1b shows how the location of the minimum value of the decision function changes when $l_1 \neq l_2$. The number of iterations of each loop at this minimum point can be derived from Eq. (10) to be

$$r_{i, \min} = \frac{|\mathcal{T}|}{|T_i|} \left(\left(\frac{l_3 - i}{l_i} \right)^2 + 1 \right)^{-1} \quad \text{for } i = 1, 2.$$

The further the number of iterations in each loop is from this minimum value, the more favorable it will be to use the natural partitioning. When the fraction of

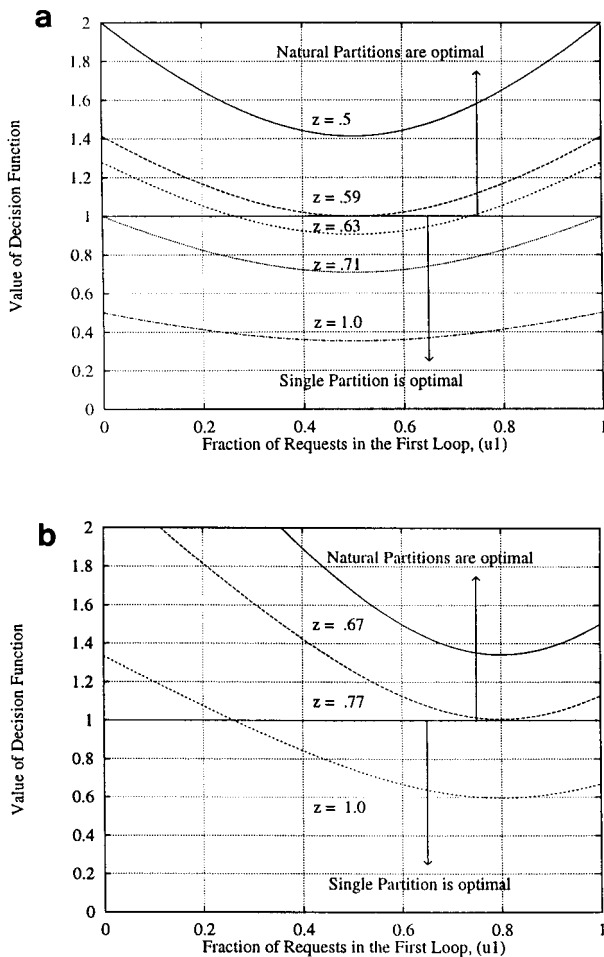


FIG. 1. Locality of two loops with common paths. (a) Loop 1 has $l_1 = l_2$ different paths; $z = l/\lambda_n$. (b) Loop 1 has $l_1 = 2l_2$ different paths; $z = l_1/\lambda_n$.

requests in the larger loop is sufficiently small, the natural partitioning is optimal regardless of the overlap between loops, similar to the statement of Lemma 2.

Table 4 provides examples that illustrate the relationships shown in Fig. 1. In summary, there is always a degree of overlap below which the natural partitioning is better than a single partition. The overlap can be exactly determined for any number of loops with equal numbers of different paths and for two loops with any numbers of different paths.

4.3. Partitioning Loops with Uniform Overlap

Theorem 4 compares the L -measure of \mathcal{P}_N and \mathcal{P}_S on sequences of paths from several loops. Except for the case of two loops, it does not identify the optimal partitioning function. There are 2^{n-1} ways to partition a sequence of n loops. To find the optimal partitioning, the decision function from Theorem 4 could be incorporated into an algorithm that determines which subsequences of loops should be combined into a single partition to obtain the optimal L -measure. Development of

TABLE 4
Different Partitionings of Overlapping Loops

Application sequence \mathcal{T}	I_1/λ_n	u_1	I_2/I_1	Partitioning	Path sets M_i	$L(\mathcal{T}, \mathcal{P})$
$(1, 2, 3, 4)^{20} (3, 4, 5, 6)^{20}$	0.67	0.5	1.0	\mathcal{P}_N	$\{1, 2, 3, 4\} \{3, 4, 5, 6\}$	3.54
				\mathcal{P}_S	$\{1, 2, 3, 4, 5, 6\}$	4.44
$(1, 2, 3, 4)^{20} (4, 5, 6, 7)^{20}$	0.57	0.5	1.0	\mathcal{P}_N	$\{1, 2, 3, 4\} \{4, 5, 6, 7\}$	3.54
				\mathcal{P}_S	$\{1, 2, 3, 4, 5, 6, 7\}$	3.27
$(1, 2, 3, 4, 5, 6, 7, 8)^{10} (1, 2, 3, 4)^{20}$	1.0	0.5	0.5	\mathcal{P}_N	$\{1, 2, 3, 4, 5, 6, 7, 8\} \{1, 2, 3, 4\}$	1.86
				\mathcal{P}_S	$\{1, 2, 3, 4, 5, 6, 7, 8\}$	2.50
$(1, 2, 3, 4, 5, 6, 7, 8)^5 (1, 2, 3, 4)^{30}$	1.0	0.25	0.5	\mathcal{P}_N	$\{1, 2, 3, 4, 5, 6, 7, 8\}, \{1, 2, 3, 4\}$	2.53
				\mathcal{P}_S	$\{1, 2, 3, 4, 5, 6, 7, 8\}$	2.50

such an algorithm is beyond the scope of this paper. Instead, we will continue to look at looping programs with specific structures in order to develop closed form solutions for locality in different cases and to gain insight into the identification of communication working sets.

In some programs it may be advantageous to combine some, but not all, loops. We can extend our investigation of partitioning functions to include those that combine several adjacent loops by considering loops with identical characteristics that overlap in a uniform manner. We again consider the case where $l_i = l$ and $r_i |T_i| = r|T|$ for all i . We will divide the sequence of loops into groups of size w and place each group of w loops together in a single partition. We will call this the “grouped-by- w ” partitioning, \mathcal{P}_{Gw} . For simplicity we assume w divides n , creating n/w partitions, each containing w loops. We define *uniform overlap* to mean that the number of different paths is the same in each group of w , so that the ratio l/λ_w is the same in all groups.

Loops with a Common Core Set of Paths

One example of a sequence of loops with uniform overlap is one in which the loops have a common core set of paths. These paths may be used for a basic function such as computing a global sum or testing a completion condition. The remaining paths in each loop are disjoint, for communication specific to the function of the loop. We denote the number of paths of these two types as l_c and l_d , respectively. The amount of overlap is thus $l/\lambda_w = l/(wl_d + l_c)$, which is the same for each group of w . The L -measure of the natural partitioning of n such loops is $r|T|/(l^2\sqrt{n})$. The L -measure from partitioning the loops into groups of w is

$$\frac{r|T|}{l^2\sqrt{n}} \left(w\sqrt{w} \left(\frac{l}{\lambda_n} \right)^2 \right).$$

$\mathcal{P}_N(\mathcal{T})$ has a greater L -measure than $\mathcal{P}_{Gw}(\mathcal{T})$ when $l/\lambda_w \leq 1/\sqrt{w\sqrt{w}}$. This is a generalization of Corollary 4.1. This decision function can also be expressed in terms of the fraction of paths that are common to all loops as $l_c/l \leq (w - \sqrt{w\sqrt{w}})/(w - 1)$. The result is shown in Fig. 2. When $l_c/l < 0.32$, the natural partitioning has a greater L -measure than any *grouped-by- w* partitioning. With any greater fraction of common paths, the L -measure is increased by combining a pair of loops into each partition. As l_c/l increases, the L -measure of grouped-by- w exceeds that of the natural partitioning for larger values of w . The value of w which has the largest L -measure can be found at the minimum point on the curve for each value of l_c/l .

For example, if the application can be represented by $(1, 2, 3, 4)^r (1, 2, 5, 6)^r (1, 2, 7, 8)^r$ then $l_c/l = 0.5$. Figure 2 shows that $L(\mathcal{T}, \mathcal{P}_{Gw})$ is greatest at $w = 3$, and the L -measure is increased by combining all three loops into a single partition. If the application is $(1, 2, 3, 4)^r (1, 5, 6, 7)^r (1, 8, 9, 10)^r$, then $l_c/l = 0.25$ and the natural partitioning has a greater L -measure.

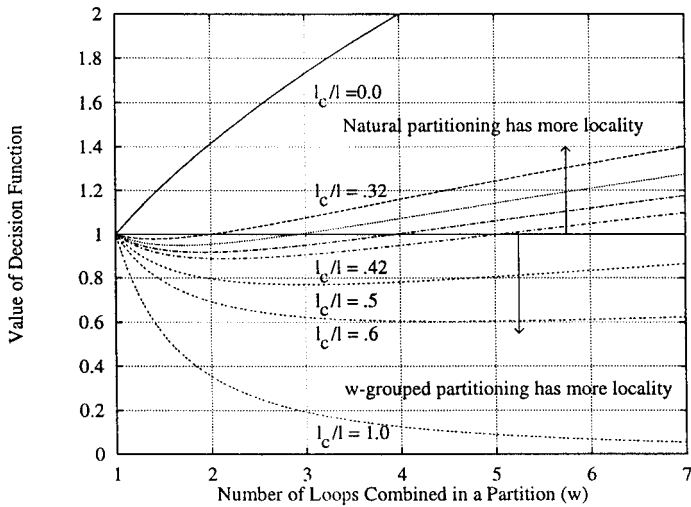


FIG. 2. Locality when overlapping natural partitions are combined.

Loops with Progressively Shifting Paths

Another example of a sequence with uniform overlap is one in which the paths used by the loops progressively shift. That is, loop i has l_d distinct paths, and shares $l_c/2$ paths with loop $i - 1$ ($i > 1$) and $l_c/2$ paths with loop $i + 1$ ($i < n$). It differs from the previous example in that loops that are not adjacent do not have any paths in common. This sequence has a uniform overlap of $\lambda_w = w(l_d + l_c/2) + l_c/2$. While the decision function in terms of λ_a/l is the same as for the previous example, the relationship between the natural and grouped-by- w partitionings is different when expressed in terms of l_c/l . The decision function for this example is $l_c/l \leq 2(w - \sqrt{w \sqrt{w}})/(w - 1)$, which is greater than 1 for $w \geq 16$. For these values of w , the natural partitioning will always have an L -measure greater than a grouped-by- w partitioning.

Admissible Partitioning Functions

We can compare the natural partitioning to a broader set of partitioning functions if we describe further what is required of functions that identify working sets. Such functions should recognize communication patterns and construct, in a consistent manner, partitions that represent working sets. We define functions that meet the consistency criteria to be *admissible* partitioning functions.

We can see the value of a consistency requirement by considering the previous examples of sequences of loops with identical locality characteristics and uniform overlap. One possible partitioning of such sequences is to combine the first two loops into a single partition and to place each remaining loop into its own partition. In some cases, such a partitioning produces an L -measure greater than that of the natural partitioning. However, we would not normally identify the working sets of a program in such an inconsistent manner. When the characteristics of all

loops and the relationships between all loops are identical, the working sets should consist of either two loops or one loop, but not sometimes one and sometimes two.

We can formalize the consistency requirement for an admissible partitioning function in these cases. Consider the natural partitioning of the sequence $\mathcal{T} = T_1^{r_1} \dots T_n^{r_n}$, and let P_i be the natural partition of loop i . Let $P_i \dots P_j$ be a sequence of partitions which are combined into a single partition by the partitioning function \mathcal{P} , so that partition boundaries occur after P_{i-1} and P_j . Let $P_{i'} \dots P_{j'}$ be a sequence disjoint from $P_i \dots P_j$ with an identical number of partitions ($j-i = j'-i'$). If $\mathcal{P}(\mathcal{T})$ has a partition boundary after both $P_{i'-1}$ and $P_{j'}$, and \mathcal{P} is admissible, then for all $k, i' \geq k < j'$, P_k does not precede a partition boundary of $\mathcal{P}(\mathcal{T})$.

A consequence of this definition is that for some sequences of n loops, some of the 2^{n-1} possible partitionings cannot be produced by an admissible partitioning function. In the case of uniformly overlapping loops, if \mathcal{P} combines one subsequence of w natural partitions into a single partition, then $\mathcal{P}(\mathcal{T})$ cannot contain any subsequences of w natural partitions. All such subsequences must be combined in some way, subject to the consistency requirement.

We can extend the results for this application to show that when the overlap is low, the natural partitioning is optimal for any admissible partitioning function. We first note that in this case there is a limit to the number of partitions that can be produced by an admissible function.

LEMMA 5. *Consider a sequence of n natural partitions with uniform overlap where $l_i = l$ and $r_i |T_i| = r_j |T_j|$ for all i and j . Let \mathcal{P} be an admissible partitioning function other than \mathcal{P}_N . Then \mathcal{P} has no more than $2n/3$ partitions.*

Proof. Since \mathcal{P} is admissible and $\mathcal{P} \neq \mathcal{P}_N$, it must combine w loops into a partition for some value of $w > 1$. The longest subsequence of loops that are not partitioned together must be less than or equal to $w - 1$. Thus, every subsequence of $2w - 1$ loops is placed in no more than w partitions. The number of partitions cannot exceed $wn/(2w - 1)$. Since $w \geq 2$, the maximum number of partitions is $2n/3$. ■

We can now construct a decision function to determine when the natural partitioning has greater locality than any other admissible partitioning for the application with core paths.

THEOREM 6. *Let $\mathcal{T} = T_1^{r_1} T_2^{r_2} \dots T_n^{r_n}$ be a sequence of loops with l_c paths in common and l_d distinct paths, and let \mathcal{P} be an admissible partitioning function. Let $l_i = l$ and $r_i |T_i| = r_j |T_j|$ for all i and j . Then $L(\mathcal{T}, \mathcal{P}_N) \geq L(\mathcal{T}, \mathcal{P})$ when $l_c/l \leq 0.14$.*

Proof. Let p be the number of partitions created by \mathcal{P} and note that, for each group of w natural partitions $w/\lambda_w < 1/l_d$. From the decision function, we can determine that a sufficient condition for the natural partitioning to have a greater L -measure is that

$$d\left(\frac{p}{n}, \frac{l_c}{l}\right) = \sqrt{\frac{n}{p}} \left(1 - \frac{l_c}{l}\right) \left(1 - \frac{l_c}{l} \left(1 - \frac{p}{n}\right)\right) \geq 1.$$

For any overlap l_c/l , this function reaches its minimum value at $p/n = (1 - l_c/l)/(l_c/l)$. For an admissible function, $p/n \leq 2/3$. Thus, the minimum value is

TABLE 5
Partitioning Multiple Overlapping Loops

	Looping program	Alternative partitioning	Decision function	\mathcal{P}_N partitioning is always better
Any		\mathcal{P}_S	$\frac{\lambda_n^2}{\sum_1^n l_i} \sqrt{\sum_1^n \frac{u_i^2}{l_i^2}} \geq 1$	$\lambda_n^2 \geq \left(\sum_1^n l_i \right) \sqrt{\sum_1^n l_i^2}$
	Any n	\mathcal{P}_S	$\frac{\lambda_n^2}{n l^2} \sqrt{\sum_1^n u_i^2} \geq 1$	$\frac{l}{\lambda_n} \leq \frac{1}{\sqrt{n} \sqrt{n}}$
	$n = 2$	\mathcal{P}_S	$\frac{\lambda_2^2}{2 l^2} \sqrt{u_1^2 + u_2^2} \geq 1$	$\frac{l}{\lambda_2} \leq .59$
For all i ,	$\left\{ \begin{array}{l} \text{Core} \\ \text{paths} \end{array} \right.$	\mathcal{P}_{G_w}	$\frac{\lambda_w}{l} \frac{1}{\sqrt{w} \sqrt{w}} \geq 1$	$\frac{l_c}{l} \leq w - \sqrt{\frac{w}{w-1}}$
$l_i = l$		\mathcal{P}_{G_2}	$\lambda_2^2 \geq 1.68$	$\frac{l_c}{l} \leq .32$
	$\left\{ \begin{array}{l} \text{Shifting} \\ \text{paths} \end{array} \right.$	\mathcal{P}_{G_w}	$\frac{\lambda_w}{l} \frac{1}{\sqrt{w} \sqrt{w}} \geq 1$	$\frac{l_c}{l} \leq \frac{2(w - \sqrt{w} \sqrt{w})}{w-1}$
		\mathcal{P}_{G_2}	$\lambda_2^2 \geq 1.68$	$\frac{l_c}{l} \leq .64$
	Uniform overlap	$\mathcal{P}_{\text{admissible}}$	$\sqrt{\frac{n}{p}} \left(1 - \frac{l_c}{l} \right) \left(1 - \frac{l_c}{l} \left(1 - \frac{p}{n} \right) \right) \geq 1$	$\frac{l_c}{l} \leq .14$

reached with the maximum number of partitions as long as $l_c/l \leq 0.6$. In this range of overlap, $d(p/n, l_c/l) \geq 1$ whenever $l_c/l \leq 0.14$. Thus, for overlap less than this amount, the natural partitioning has an L -measure greater than that of any other admissible partitioning. ■

Table 5 summarizes the results so far. This table shows when the natural partitioning is better at identifying working sets than alternative partitioning functions. This occurs when the amount of overlap between loops is low. The exact amount depends on the application and the alternative partitionings considered.

5. THE NATURAL LOCALITY OF LOOPING SEQUENCES

We have used our measure to find that placing a loop into a single partition often identifies the working set. We have investigated when the working sets in a sequence of loops consist of the paths from each loop in cases where loops are disjoint and where loops overlap. Since the problem of finding the optimal partitioning can be complex, we have looked at different approaches that combine several loops into a single working set. While the measure can be applied to any partitioning, it can be hard to determine an optimal partitioning function in the general case.

Since the natural partition can be the best of the alternatives even when loops overlap, we will estimate the locality of a sequence of loops to be the L -measure of its natural partitioning. We expect this to be a reasonable approximation, especially for simple cases. We define the *natural locality* of a sequence of requests from a looping program to be $\mathcal{L}_N(\mathcal{T}) = L(\mathcal{T}, \mathcal{P}_N)$. We will approximate the locality of a sequence using its natural locality, $\mathcal{L}(\mathcal{T}) \approx \mathcal{L}_N(\mathcal{T})$.

We consider three final points. First, we look for general properties of locality based on our measure. Second, we validate the locality approximation by comparing its results to a model of communication performance. Third, we briefly look at how our results would be affected by an alternative measure for spatial locality.

5.1. Properties of the Locality Measure

Using the natural partitioning, we can develop properties that partially order communication request sequences by locality. These properties hold for the natural locality $\mathcal{L}_N(\mathcal{T})$, and for locality $\mathcal{L}(\mathcal{T})$ when loops have a sufficient number of iterations.

Property L1. Given two disjoint sequences T_1 and T_2 and a fixed number of iterations r ,

$$\mathcal{L}(T_1^r T_2^r) > \mathcal{L}((T_1 T_2)^r).$$

Locality always increases with disjoint sets of paths in separate partitions.

Property L2. Given two loops, T_1^r and T_2^r , where $r_1|T_1| = r_2|T_2|$ and $l_1 < l_2$,

$$\mathcal{L}(T_1^r) > \mathcal{L}(T_2^r).$$

For an equal number of requests, a loop with fewer paths will have greater locality than a loop with more paths.

Property L3. Given two sequences T_1 and T_2 with $l_1 < l_2$, and iteration values r_1, r_2, r'_1 , and r'_2 , where $r_1 > r'_1 > r_{1, \min}$ and $r_1|T_1| + r_2|T_2| = r'_1|T_1| + r'_2|T_2|$,

$$\mathcal{L}(T_1^{r_1} T_2^{r_2}) > \mathcal{L}(T_1^{r'_1} T_2^{r'_2}).$$

When loops have a different number of paths, the distribution of requests between two loops affects locality. The minimum locality occurs when most of the requests come from the loop with the largest number of different paths. When the fraction of requests from the loop with the fewest paths increases, locality increases.

Table 6 shows the locality of several example communication patterns with the same number of requests. These sequences were chosen to have different looping structures and to have loops with a variety of locality parameters l , $|T|$, and r . Clearly, the ordering obtained from the L -measure and shown in Table 6 follows the intuitive notion of locality. For example, $(1, 2)^{80}$ is intuitively more local than $(1, 2)^{40} (3, 4)^{40}$, which is more local than $(1, 2, 3, 4)^{40}$.

5.2. Locality and Communication Performance

While the measure developed in this paper measures the communication locality of an application's requests, a more important program characteristic is its communication performance. Communication performance depends on the characteristics of the network on which the program is run, while locality does not. One way to validate the locality measure is to see how well it predicts communication performance.

In [19], we describe a model of the communication delays encountered in a circuit-switched network which is controlled by time division multiplexing. The model applies to compiled communication, in which a sequence of network states is predetermined by the compiler to provide the paths needed by the program. To adapt to changes in working set, these states are loaded into the network hardware as the program executes. Additionally, time division multiplexing can be used

TABLE 6

L-measure of Several Sequences of Equal Length

Sequence \mathcal{T}	$L(\mathcal{T}, \mathcal{P})$	Sequence \mathcal{T}	$L(\mathcal{T}, \mathcal{P})$
$(1, 2)^{80}$	40.0	$(1, 2, 3)^{20} (4, 5, 6, 7)^{20}$	6.28
$(1, 1, 1, 2)^{40}$	40.0	$(1)^{35} (2, 3)^{25} (4, 5, 6, 7, 8)^{15}$	5.69
$(1, 2, 3, 3)^{40}$	17.8	$(1, 2, 3, 4, 5, 6)^{27}$	4.44
$(1, 2)^{40} (3, 4)^{40}$	14.1	$(1, 2, 3, 4)^{30} (5, 6, 7, 8)^{10}$	3.95
$(1, 2, 3, 4)^{40}$	10.0	$(1, 2, 3, 4)^{20} (5, 6, 7, 8)^{20}$	3.54
$(1)^{70} (2, 3, 4, \dots, 9, 10)^{10}$	7.07	$(1, 2, 3, 4, 5, 6, 7, 8)^{20}$	2.50
$(1, 2, 3, 4, 5)^{32}$	6.40	$(1, 2, 3, 4, \dots, 9)^{18}$	1.96
$(1, 2)^{50} (3, 4, 5, 6, 7, 8)^{10}$	6.37	$(1, 2, 3, 4, \dots, 10)^{16}$	1.60

to cycle through a sequence of K states which have been loaded together. While reducing the number of times the network state is loaded into the hardware, the additional paths in the cycle may delay the program's use of a path.

Network performance characteristics include the time to load the network state into the hardware (E), the cycle time of states in the multiplexed sequence (A), and the number of paths the network can provide simultaneously (m). The delay per request can be described in terms of A , E , m , and the multiplexing degree K . Analogous to the natural partitioning, a multiplexing degree $K = \lceil l/m \rceil$ contains all the paths required by a loop and is defined to be *one-partition multiplexing*. The associated delay, i.e., the cost of communication, is shown in [19] to be

$$\text{Cost}(T) = E + \frac{AKr|T|}{2m}.$$

This cost is less than the cost of any other multiplexing degree when $2|T| < Em/A < r|T|/2$. This shows how the choice of optimal multiplexing degree can be determined from the relationship between network characteristics and application characteristics.

Communication locality should clearly affect performance using such a network. The size of the working set affects the multiplexing degree K . The amount of temporal locality affects the frequency with which a new state must be loaded. Thus, it is reasonable to validate the locality model by comparing the locality of a sequence of requests to the cost of compiled communication for those requests in a circuit switched network.

Corresponding to the locality properties we can develop cost properties for partially ordering applications when one-partition multiplexing is used for each loop. Equality or inequality in these properties is dependent on the network's value of m . In general, looping sequences with disjoint sets of paths and a sufficient number of iterations will be placed in the same partial order by both locality and cost.

Property C1. Given two disjoint sequences T_1 and T_2 , and a fixed number of iterations r ,

$$\text{Cost}(T_1^r T_2^r) \leq \text{Cost}((T_1 T_2)^r).$$

Cost cannot increase with disjoint sets of paths in separate partitions. Separate partitions may decrease cost even when there is some overlap between the sets of paths.

Property C2. Given two loops, $T_1^{r_1}$ and $T_2^{r_2}$, where $r_1|T_1| = r_2|T_2|$ and $l_1 < l_2$,

$$\text{Cost}(T_1^{r_1}) \leq \text{Cost}(T_2^{r_2}).$$

For an equal number of requests, a loop with fewer paths cannot cost more than a loop with more paths.

Property C3. Given two sequences, T_1 and T_2 , and iteration values r_1 , r_2 , r'_1 , and r'_2 , where $r_1 > r'_1$, $l_1 < l_2$, and $r_1|T_1| + r_2|T_2| = r'_1|T_1| + r'_2|T_2|$,

$$\text{Cost}(T_1^{r_1} T_2^{r_2}) \leq \text{Cost}(T_1^{r'_1} T_2^{r'_2}).$$

When loops have a different number of paths, the distribution of requests between two loops affects cost. As the proportion of requests in the loop with fewer paths increases, cost cannot increase.

Figure 3 shows the relative cost and locality of the sequences in Table 6. In order to make cost and locality trends similar, the inverse of locality is plotted in Fig. 3a. The value of Em/A has very little effect on the relative costs in Fig. 3. For these sequences, the statistical correlation between cost and locality is 0.93 and locality is a good predictor of communication cost.

To broaden the variety of sequences on which locality and cost are compared, we randomly generated sets of parameters for use with the two models. Locality was computed using the natural partitioning, and cost was computed using one-partition multiplexing. The parameters were chosen to be representative of characteristics that could be found in parallel programs:

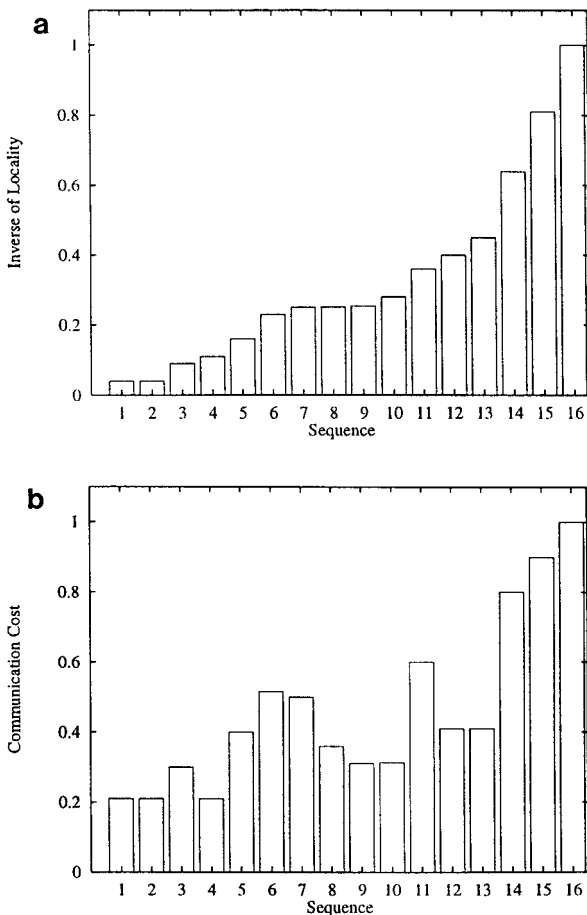


FIG. 3. Locality as a predictor of relative communication cost. (a) Inverse of locality for sequences in Table 6. (b) Relative cost of sequences in Table 6.

- $|T_i|$ randomly chosen between 32 and 960;
- l_i randomly chosen between 32 and $|T_i|$;
- r_i randomly chosen between 3000 and 3500;
- n randomly chosen between 1 and 20 loops.

Sequences were generated in sets of 1000 using various additional restrictions on the above parameters. The correlation between cost and locality for each set was consistently in the range of 0.60 to 0.80, with even higher correlations for sequences from a single loop. This high degree of correlation over a wide range of sequences suggests that using the natural partitioning to identify working sets can be a useful way to predict communication performance.

5.3. Communication Patterns in Parallel Programs

Communication in parallel programs often involves permutation patterns. These patterns arise from the algorithmic decomposition of a problem and subsequent assignment of tasks to processors. This decomposition often reflects a logical arrangement of processors with characteristic communication patterns. These patterns are reflected in the analysis of many benchmarks. See, for example, the locality analysis in [5].

For example, it is common in programs that use a Jacobi algorithm for finite element analysis to view processors as logically arranged in an array of one or more dimensions. (This may or may not be the physical arrangement of processors.) With this logical view, communication is often from each processor to its nearest neighbors along each dimension of the array. These permutations are disjoint. Nearest neighbor communication is also used in parallel programs that perform matrix multiplication. Systolic applications are often associated with arrays where communication occurs in only one direction along each dimension.

The transposition of a matrix can be implemented in a single permutation when each processor holds a block of the matrix. The more general case of data redistribution may require several permutations, depending on the specific requirements.

Another common logical arrangement of N processors is as a $\log N$ -dimensional binary hypercube. Communication across each dimension of the hypercube is a permutation, and the permutations for all dimensions are disjoint. The fast Fourier transform uses these permutations, as do some sorting algorithms. An efficient all-to-all communication pattern can be implemented by using these permutations in sequence.

The models of the previous sections can easily be applied to sets of disjoint permutations such as these. In a nonblocking network where m equals the number of processors, we can interpret each integer in the locality notation to be not just a single path, but the entire set of paths required for the permutation. A loop requiring m paths in each of two network states can therefore be represented as $(1, 2)^r$. This loop can be contained in a single multiplexed network state with a multiplexing degree of two. For example, consider m processors arranged as a $\log m$ -dimensional binary hypercube. The integer i can represent the set of m paths used for communication over dimension i . These sets of paths are disjoint. The notation $(1, 2, \dots, \log m)^r$ then represents the pattern of m processors communicating over

TABLE 7
Partitioning Relationships with an Alternative Locality Measure

		\mathcal{P}_N is always better	
	Alternative partitioning	Base measure	Alternative measure
Looping program		$\sqrt{\frac{\left(\frac{ T_i }{ M_i }\right)^2}{\sum M_i }}$	$\sqrt{\frac{\left(\frac{ T_i }{ M_i }\right)^2}{\sum M_i }}$
One loop, distinct paths	$\mathcal{P}_{\text{arbitrary}}$	$r^3 \geq l$	$r \geq l$
Overlapping loops	$n = 2$	$\frac{l_1}{\lambda_n} \leq \left(\sqrt{\left(1 + \frac{l_2}{l_1}\right) \sqrt{1 + \left(\frac{l_2}{l_1}\right)^2}} \right)^{-1}$	$\frac{l_1}{\lambda_n} \leq \left(\sqrt{1 + \left(\frac{l_2}{l_1}\right)^2} \right)^{-1}$
	$n = 2, l_1 = l_2 = l$	$\frac{l}{\lambda_n} \leq 0.59$	$\frac{l}{\lambda_n} \leq .71$
	Any $n, l_i = l$, core paths	$\frac{l_c}{l} \leq \frac{2W - \sqrt{W} \sqrt{W-1}}{W-1}$	$\frac{l_c}{l} \leq \frac{W - \sqrt{W}}{W-1}$
	$n = 3, l_i = l$, core paths	$\frac{l_c}{l} \leq 0.36$	$\frac{l_c}{l} \leq 0.63$

each dimension in succession in a loop executed r times. Similarly, $(1, 2, 3, 4)^r$ can represent a pattern where each of m processors in a two-dimensional torus communicates once to its nearest neighbors in each dimension.

5.4. An Alternative Locality Measure

The locality measure contains an implicit balance of the spatial and temporal locality of a sequence of requests. To understand how changing this balance affects our results, we replaced the measure of spatial locality from Eq. (1) with one that reduces the locality of large working sets. A locality measure built with this should identify working sets with a smaller number of different paths.

Specifically, we constructed a measure of spatial locality using the same mean-root-square approach that was used for constructing the measure of temporal locality. This alternate measure of spatial locality is defined as

$$\text{Spatial Locality}_{\text{Alt}}(\mathcal{T}, \mathcal{P}) = \frac{1}{(1/n) \sqrt{\sum_{i=1}^n |M_i|^2}}.$$

Lemmas 1 and 2 and Theorem 3 can be shown to hold for the locality measure built from this definition. Table 7 shows how the results for different sequences are affected. The alternative measure does encourage the identification of smaller working sets. While the decision functions retain the same general form, the conditions under which grouping increases locality become more restrictive. On the one hand, the number of iterations required to combine paths from a loop into a single partition increases from $r > l^{1/3}$ to $r > l$. On the other hand, the overlap in a sequence of loops must be greater before combining two or more loops into a single partition increases locality. For example, combining two loops with an equal number of different paths into a single partition increases locality using the base measure when each loop has more than 59% of the total number of paths. With the alternative measure, the locality of the natural partitioning remains the greatest until each loop has more than 71% of the total paths.

6. CONCLUSIONS

Communication locality is a characteristic of an application that, in some cases, can be understood before the application is executed. For example, with compiled communications we can obtain the sequence of communication requests in advance of program execution. Intuitively, communication locality should affect the demands on the network and hence communication performance as well. We proposed a model for measuring the locality of parallel applications and showed that the results of applying this measure correspond to our intuitive notions about locality. We found that it is hard to determine the optimal partitioning of a sequence of requests, even for restricted cases of looping sequences. We argued that for sufficiently disjoint loops and a sufficiently large number of iterations, the natural partitioning of an application provides the greatest measure of locality of any partitioning, and communication working sets can be identified by loop boundaries.

While the model provides an objective basis for determining the working set of communication paths, the real usefulness of the model is the correspondence between locality and cost. We compared the locality results to those of a separately developed model of communication performance for an environment where communication locality is expected to be important. We found that to a high degree, increased locality correlates with reduced communication cost. Thus, the locality measure can be used as a tool to improve performance of an application without specific knowledge of the network over which it will communicate. The precise effect of locality on cost depends on many factors such as the network parameters, the order of the communication requests and the number of paths used. The programmer can use locality to take expected communication performance into consideration when choosing the most appropriate algorithm to solve a specific problem. Locality can also be used to evaluate the impact of different data distribution alternatives on communications.

We note the similarity between replacing paths in a network state and the paging process used to manage a processor's physical memory. The memory management concepts of locality of reference and working set can be applied to path management in a circuit switched network, as well. We expect that the ability to identify and manage the working set of paths used by a parallel program will have a strong impact on communications performance in an interconnection network.

REFERENCES

1. V. Adve and M. Vernon, Performance analysis of mesh interconnection networks with deterministic routing, *IEEE Trans. Parallel Distrib. Syst.* **5** (1994), 225–246.
2. W. Athas and C. Seitz, Multicomputers: Message-passing concurrent computers, *IEEE Comput.* **21** (1988), 9–24.
3. F. Cappello and C. Germain, Toward high communication performance through compiled communications on a circuit switched interconnection network, in "Proceedings of the First IEEE Symposium on High Performance Computer Architectures," pp. 44–53, IEEE Computer Society Press, Los Alamitos, CA, 1995.
4. C. Chen, D. Agrawal, and J. Burke, DBCube: A new class of hierarchical multiprocessor interconnection networks with area efficient layout, *IEEE Trans. Parallel Distrib. Syst.* **4** (1993), 1332–1344.
5. B. Dao, S. Yalamanchili, and J. Duato, Architectural support for reducing communication overhead in multiprocessor interconnection networks, in "Proceedings of the 3rd International Symposium on High Performance Computer Architecture, HPCA '97," pp. 343–352, IEEE Computer Society Press, Los Alamitos, CA, 1997.
6. P. Denning, The working set model for program behavior, *Comm. Assoc. Comput. Mach.* **11** (1968), 323–333.
7. D. Feitelson and L. Rudolph, Coscheduling based on runtime identification of activity working sets, *Internat. J. Parallel Programming* **23** (1995), 135–159.
8. K. Grimsrud, J. Archibald, R. Frost, and B. Nelson, On the accuracy of memory reference models, in "Computer Performance Evaluation, Modeling Techniques and Tools: 7th International Conference Proceedings," pp. 369–388, Springer-Verlag, Berlin/New York, 1994.
9. N. Gulati, C. Williamson, and R. Bunt, LAN traffic locality: Characterization and application, in "Local Area Network Interconnection: Proceedings of the First International Conference," pp. 233–250, Plenum, New York, 1993.

10. M. Holliday and M. Stumm, Performance evaluation of hierarchical ring based shared memory multiprocessors, *IEEE Trans. Comput.* **43** (1994), 52–67.
11. K. L. Johnson, The impact of communication locality on large-scale multiprocessor performance, *Comput. Architecture News* **20** (1992), 392–402.
12. D. Lahaut and C. Germain, Static communications in parallel scientific programs, in “PARLE ’94 Parallel Architecture and Languages—Europe,” Springer-Verlag, New York/Berlin, 1994.
13. P. Lee, Efficient algorithms for data distribution on distributed memory parallel computers, *IEEE Trans. Parallel Distrib. Syst.* **8** (1997), 337–352.
14. J. Li and M. Chen, Compiling communication-efficient programs for massively parallel machines, *IEEE Trans. Parallel Distrib. Syst.* **2** (1991), 361–375.
15. D. T. Michel and W. C. Hobart, Jr., Toward a unified model of program behavior, *Perform. Eval.* **20** (1994), 27–44.
16. R. Numrich, P. Springer, and J. Peterson, Measurement of communication rates on the Cray T3D interprocessor network, in “Proceedings of HPCN Europe ’94,” Springer-Verlag, Berlin/New York, 1994.
17. Y. Pan and Y. Chuang, Properties and performance of the block shift network, *IEEE Trans. Circuits Syst. Fund. Theory Appl.* **44** (1997), 93–102.
18. V. Peris, M. Squillante, and V. Naik, Analysis of the impact of memory in distributed parallel processing systems, *Perform. Eval. Rev.* **22** (1994), 5–18.
19. C. Salisbury and R. Melhem, Modeling communication costs in multiplexed optical switching networks, in “11th International Parallel Processing Symposium (IPPS),” pp. 71–79, IEEE Computer Society Press, Los Alamitos, CA, 1997.
20. A. Sivasubramaniam, A. Singla, U. Ramachandran, and H. Venkateswaran, Abstracting network characteristics and locality properties of parallel systems, in “Proceedings of the First IEEE Symposium on High Performance Computer Architecture,” pp. 54–63, IEEE Computer Society Press, Los Alamitos, CA, 1995.
21. D. Thiébaud, J. Wolf, and H. Stone, Synthetic traces for trace-driven simulation of cache memories, *IEEE Trans. Comput.* **41** (1992), 388–410.
22. X. Yuan, R. Melhem, and R. Gupta, Compiled communication for all-optical TDM networks, in “Supercomputing ’96,” IEEE Computer Society Press, Los Alamitos, CA, 1996.

CHARLES SALISBURY received the B.S. and M.S. degrees from the University of Wisconsin-Madison. He was employed by IBM in marketing, system performance evaluation, and software development. He received M.S. and Ph.D. degrees in Computer Science from the University of Pittsburgh in 1995 and 1998, respectively. His research interests include parallel computing, interconnection networks, optical networking, and communication switching.

ZHIXIONG CHEN received an M.S. in Computer Science and a PH.D. in Applied Mathematics from the University of Pittsburgh in 1997. He is currently on the technical staff at IBM-TRANSARC. His research interests are distributing computing algorithms and neuronal modeling.

RAMI MELHEM received a B.E. in Electrical Engineering from Cairo University in 1976, an M.A. degree in Mathematics and an M.S. degree in Computer Science from the University of Pittsburgh in 1981, and a Ph.D. degree in Computer Science from the University of Pittsburgh in 1983. He was an Assistant Professor at Purdue University prior to joining the faculty of the University of Pittsburgh in 1986, where he is currently a Professor of Computer Science and Electrical Engineering. His research interest include fault-tolerant and real-time systems, optical interconnection networks, high performance computing and parallel computer architectures. Dr. Melhem served on program committees of numerous conferences and workshops on parallel, distributed and fault-tolerant systems. He served as the general chair for the *Third International Conference on Massively Parallel Processing Using Optical Interconnections* and he is on the steering committee of that conference. He was on the editorial board of the *IEEE Transactions on Computers* and served on the advisory boards of the IEEE technical committees on parallel processing and computer architectures. He is the editor for the Plenum Book Series in Computer Science and is on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems*. Dr. Melhem is a senior member of IEEE and a member of the Association for Computing Machinery.