The Effect of Multiprocessor Radius on Scaling

Patrick H. Worley
THE EFFECT OF MULTIPROCESSOR RADIUS ON SCALING

Patrick E. Worley
Oak Ridge National Laboratory
Mathematical Sciences Section
P.O. Box 2009, Bldg. 9207-A
Oak Ridge, TN 37831-8083

Date Published: June, 1990

Research was supported by the
Applied Mathematical Sciences Research Program
of the Office of Energy Research,
U.S. Department of Energy.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC-05-84OR21400
Contents

1 Introduction ........................................... 1

2 Assumptions ........................................... 2
   2.1 Multiprocessor assumptions ......................... 2
   2.2 Algorithm model ................................... 3
   2.3 Problem assumptions I-V ........................... 5

3 Algorithm-dependent results ......................... 7
   3.1 Information-theoretic lower bounds ................. 7
   3.2 Example communication bound algorithms .......... 10

4 Problem-dependent results .......................... 12
   4.1 Communication bound problems ...................... 12
   4.2 Problem-dependent bounds ......................... 13
   4.3 Sufficient conditions for a problem to be communication bound ........ 17

5 Scaling results ...................................... 19
   5.1 Problem assumption VI ............................. 19
   5.2 Problem-scaling bound ............................. 20
   5.3 Effect of r(p) on the asymptotic growth of the parallel cost ........ 20

6 Conclusions ......................................... 22
THE EFFECT OF MULTIPROCESSOR RADIUS ON SCALING

Patrick H. Worley

Abstract

In earlier work, it was established that, for a large class of linear partial differential equations (PDEs), increasing the problem size necessarily increases the execution time, independent of the algorithm and the number of processors used to solve the problem. In this paper, the analysis is extended to take into account the effect of the radius of the multiprocessor interconnection network on the growth in the execution time.

Define \( r(p) \) to be the minimum radius over all subsets of \( p \) processors in a multiprocessor. An information-theoretic analysis is used to show that \( r(p) \) determines a lower bound on the communication cost of a parallel algorithm, and that this in turn determines a lower bound on the parallel execution time. Assume that \( r(p) \geq \beta \cdot p^3 - \mu \) for positive constants \( \beta, \gamma, \) and \( \mu \) for a given multiprocessor. For example, this type of lower bound on \( r(p) \) holds for a multiprocessor whose interconnection topology is a \( k \)-dimensional array. It is then established that, for the given class of PDEs, the time spent on interprocessor communication will be the dominant constraint on the performance of optimal algorithms when the problem and the multiprocessor are large. Moreover, as the problem and the multiprocessor increase in size, it is shown that the asymptotic increase in the parallel execution time will be determined by the communication cost and not by the computational requirements. The restriction to linear PDEs is not necessary, and similar results can be obtained for many problems in scientific computation.
1. Introduction

Current research into parallel processing and multiprocessors is driven by the need to increase computing power. Two goals are achieved by increasing computing power: more problems can be solved in a given interval of time, and problems whose solutions have heretofore been too costly to calculate can be solved.

In this paper we examine an issue related to how effective multiprocessors are at achieving these goals, namely, the issue of scaling. A multiprocessor scales if increasing the number of processors enables it to solve larger problems efficiently. A lack of parallelism in the algorithms or high communication cost can prevent this. In previous work \[23, 24\], we established that, for a large class of linear partial differential equations (PDEs), increasing the problem size (unboundedly) necessarily increases the execution time (unboundedly), independent of the algorithm and of the number of processors used to solve the problem. In this paper, we extend this analysis to describe the effect of the radius of the interconnection network of the multiprocessor on this growth in the execution time.

Define \( r(p) \) to be the minimum radius over all subsets of \( p \) processors in a multiprocessor, and assume that \( r(p) \geq \beta \cdot p^\gamma - \mu \) for positive constants \( \beta, \gamma, \) and \( \mu \). For example, this type of lower bound on \( r(p) \) holds for a multiprocessor whose interconnection topology is a \( k \)-dimensional array. Then, for the given class of PDEs, we show that the time spent on interprocessor communication will be the dominant constraint on the performance of optimal algorithms when the problem and multiprocessor sizes are large. Moreover, we show that the asymptotic rate of increase in the parallel execution time as the problem size and the number of processors increase is determined by the communication cost and not by the computation cost.

Many authors have discussed the effect of the interconnection network of a multiprocessor on the communication costs incurred when using a multiprocessor \[2, 3, 12, 15, 19\]. Similarly, other authors have discussed the effect of communication costs on the execution time of parallel algorithms for high level descriptions of discrete problems \[1, 7\], for specific discrete problems \[9\], and for implementations of specific algorithms for discrete problems \[10, 11, 13, 14, 21, 25\]. The results in this paper differ from previous work in that we derive both upper and lower bounds on the performance of optimal algorithms for continuous problems, and are able to show how the performance scales as the problem size and the number of processors scale. The restriction of this work to the given class of linear PDEs is only for the sake of clarity and concreteness. Similar arguments hold for many continuous problems arising in mathematical physics.

The outline of this paper is as follows. We describe our multiprocessor, algorithm, and problem assumptions in §2. In §3 we describe how the multiprocessor radius function constrains the performance of parallel algorithms as a function of simple information-theoretic algorithm-independent parameters. In §4 we describe how the multiprocessor radius function constrains the performance of optimal parallel algorithms for the numerical approximation of a given linear PDE. In §5 we use these results to show how the multiprocessor radius function causes the execution time of optimal parallel algorithms to grow as the size of a linear PDE problem grows. We briefly summarize our results and make concluding remarks in §6.
2. Assumptions

Our focus in this paper is on MIMD multiprocessors [8] and on modelling parallelism at the level of concurrent execution of floating point operations. This viewpoint is reflected in the following multiprocessor and algorithm models. As with our selection of problem class, we restrict our analysis to the following models for the sake of clarity and concreteness. The results will be similar for other realistic multiprocessor and algorithm models.

2.1. Multiprocessor assumptions

Multiprocessor model. We model a multiprocessor as a directed graph \((V, E)\). Each vertex \(v_i \in V\) represents a serial processor. Each edge \(e_j \in E\) represents a unidirectional communication channel in the multiprocessor. We assume that all floating point operations are computed by the composition of operators from some given set of primitive binary and unary floating point operators, and that the execution of a primitive operator is not spread over multiple processors. We also assume that addition is the fastest binary floating point operator. The parameter \(t\) refers to the minimum time required to send a single floating point number between two distinct processors in a given multiprocessor, which we call the minimum transmission time. The parameters \(f(+)\), \(f(+\rangle\), and \(f(/)\) refer to the minimum times required to add or subtract, multiply, and divide, respectively, two floating point numbers in a given multiprocessor.

Multiprocessor radius function. Define the distance from vertex \(v_1\) to vertex \(v_2\), \(D(v_1, v_2)\), to be the minimum amount of time it takes to send a single floating point number from the processor represented by \(v_1\) to the processor represented by \(v_2\). Define a center of a subset of vertices of the graph, \(V' \subseteq V\), to be a vertex in \(V'\) that minimizes the maximum distance between itself and other vertices in the subset. That is, if \(c\) is a center of \(V'\), then

\[
\max_{w \in V'} D(c, w) = \min_{v \in V'} \max_{w \in V'} D(v, w).
\]

Define the radius of the subset to be this distance,

\[
\text{radius}(V') = \min_{v \in V'} \max_{w \in V'} D(v, w).
\]

For a given multiprocessor, consider a subset of \(p\) processors with minimum radius over all such subsets of size \(p\). Define the function \(r(p)\) to be the radius of this subset. We refer to \(r(p)\) as the multiprocessor radius function. We will use this function to establish lower bounds on communication costs in parallel algorithms.

Example architectures. Most multiprocessor architectures currently in use have fairly simple graphs, with essentially homogeneous processor and communication capabilities [5], [6], [20]. The following examples are common designs, each of whose behavior is representative of a class of architectures. All of the examples can be modeled by undirected graphs: if an edge exists
from \( v_i \) to \( v_j \), then an edge also exists from \( v_j \) to \( v_i \). Additionally, all processor and communication channel capabilities are the same.

- **clique.** The graph of the architecture is a clique; that is, each processor is directly connected to every other processor. The radius of any subset of the multiprocessor containing more than one processor is \( t \).

- **k-dimensional array.** The graph of the architecture is a \( k \)-dimensional array. Each processor is connected to up to \( 2 \cdot k \) other processors. The multiprocessor radius function \( r(p) \) is never smaller than \( k \cdot t \cdot (p^{1/k} / 2 - 1) \).

- **hypercube.** The graph of the architecture is a binary hypercube. Thus, there are \( P = 2^k \) processors, where \( k \) is some nonnegative integer, and each vertex is a corner of a \( k \)-dimensional array. Each processor is connected to \( \log_2 P \) other processors. The multiprocessor radius function is \( r(p) = t \cdot \lceil \log_2 p \rceil \).

2.2. Algorithm model

**Serial algorithms.** We define an algorithm to be a partially ordered set of instructions of the form

\[
y = \text{flop}(z_1, \ldots, z_n),
\]

where \( \text{flop} \) is a floating point operator, \( y \) is a floating point variable, and \( \{z_1, \ldots, z_n\} \) are floating point constants and variables. If a floating point variable is used by two different instructions, and if one of the instructions changes the value of that variable, then the partial order specifies a precedence relationship between them. These are the only relationships established by the partial order.\(^1\)

Denote a given algorithm by \( a \). We define the serial cost of \( a \) to be the time spent executing the instructions in \( a \) on some standard serial processor, where the standard processor is assumed to satisfy the assumptions made in the previous section about the processors in the multiprocessor. We refer to this value as \( C_a \). All sequential orderings of the instructions of an algorithm that are consistent with the partial ordering are assumed to have the same serial cost. Therefore, we also refer to the partially ordered set of instructions as a serial algorithm.

**Parallel algorithms.** A parallel implementation of an algorithm on a multiprocessor specifies when and on which processor each instruction is executed, and what communication takes place during the execution of the algorithm. We will refer to this information as the scheduling of the algorithm. Define a scheduling to be well-defined if it is compatible with the partial order's precedence relationships, and if all demands made on the processors and communication channels are within their capabilities. We define a parallel algorithm to be a triple consisting of a serial algorithm, a multiprocessor architecture, and a well-defined scheduling. We refer to

\(^1\)This model ignores many of the details usually found in real algorithms. In particular, integer arithmetic and instructions controlling conditional execution are not represented. But the time spent in floating point operations generally dominates the total execution time of serial algorithms for solving PDEs numerically. Moreover, much of the other work tends to increase proportionally with the number of floating point operations, and can be "included" in the model by increasing the execution time of the floating point operations.
a given parallel implementation of an algorithm \( a \) by \(  \bar{a} \). Conversely, \( a \) is the serial algorithm associated with a given parallel algorithm \( \bar{a} \).

We define the parallel cost of a parallel algorithm \( \bar{a} \) to be the time it takes to execute the algorithm on the specified multiprocessor. We refer to this value by \( T_{\bar{a}} \). There are two distinct costs associated with a parallel algorithm: computation cost \( (C_{\bar{a}}) \) and communication cost \( (W_{\bar{a}}) \). The computation cost is the amount of time during which at least one of the processors is busy executing the instructions of the corresponding serial algorithm. If \( P \) processors are used by a parallel algorithm to execute instructions, then the computation cost is bounded from below by

\[
C_{\bar{a}} \geq \frac{C_{a}}{P},
\]

where the serial cost in this expression is based on using the "fastest" of the \( P \) processors as the standard serial processor. The communication cost is the amount of time during which at least one processor is actively sending or receiving a message or at least one of the communication channels is busy transmitting a message. Both the computation cost and the communication cost represent lower bounds on the parallel cost:

\[
\max\{C_{\bar{a}}, W_{\bar{a}}\} \leq T_{\bar{a}}.
\]

We define an optimal parallel implementation of an algorithm on a given multiprocessor to be one that minimizes the parallel cost. We define an optimal parallel algorithm to be one with an optimal parallel implementation.

**Communication bound algorithms.** If the communication cost of a parallel algorithm is the dominant component of the parallel cost, then we say that the parallel algorithm is communication bound. We make a similar definition below for a serial algorithm \( a \) and a given multiprocessor, which we will refer to as \( M \).

Consider an idealized multiprocessor \( M' \) that is constructed from \( M \) by replacing the interconnection network of \( M \) with one that is a clique and for which the transmission time over any communication channel is zero. Also assume that there is no operating system overhead in sending or receiving a message in \( M' \). Thus, any parallel implementation of \( a \) on \( M' \) will have zero communication cost.

Consider an optimal parallel implementation of \( a \) on \( M \), and call it \( \bar{a}_a \). Next, consider an optimal parallel implementation of \( a \) on \( M' \), and call it \( \bar{a}'_a \). If the parallel cost of \( \bar{a}'_a \) is less than half the parallel cost of \( \bar{a}_a \), then we say that the serial algorithm \( a \) is communication bound on \( M \). If \( a \) is communication bound, then the interprocessor communication subsystem has a larger effect on the execution time of optimal algorithms for \( M \) than do the inherent parallelism in the serial algorithm and the speed of floating point computation. If the parallel cost of \( \bar{a}'_a \) is more than half the parallel cost of \( \bar{a}_a \), then we say that the serial algorithm \( a \) is computation bound on \( M \). If \( a \) is computation bound, then the partial order of the serial algorithm and the speed of floating point computation are the dominant constraints on the performance of optimal parallel implementations on \( M \), and \( M \) is “adequate” for this algorithm.
2.3. Problem assumptions I-V

The following assumptions are commonly satisfied when numerically approximating the solution of scalar linear partial differential equations (PDEs). As stated, they also apply to a larger class of problems.

Solution operator assumption. For any nonnegative integer \( k \), let \( \mathbb{R}^k \) be the \( k \)-dimensional Euclidean vector space and let \( I_k \) be the \( k \)-dimensional unit cube,

\[
I_k = [0,1] \times \cdots \times [0,1] \subset \mathbb{R}^k.
\]

For some \( d \geq 1 \), let \( \Omega \) be a compact subset of \( \mathbb{R}^d \).

(I) We assume that we are approximating a scalar function \( u(\mathbf{z}) \) that is defined on \( \Omega \) and that can be represented by an expression of the form

\[
u(\mathbf{z}) = \sum_{i=1}^{S} \int_{I_{d_i}} \Psi_i(\mathbf{\bar{z}}, \mathbf{\bar{x}}) g_i(\mathbf{\bar{x}}) d\mathbf{\bar{x}}
\]

for all \( \mathbf{\bar{z}} \in \Omega \). The constant \( S \) is a positive integer, \( \{d_i | i \in \{1, \ldots, S\} \} \) is a set of nonnegative integers, \( \{g_i | i \in \{1, \ldots, S\} \} \) is a set of functions representing the problem data, and \( u(\mathbf{z}) \) is the solution function. For each \( i \in \{1, \ldots, S\} \) and \( \mathbf{\bar{z}} \in \Omega \), \( \Psi_i(\mathbf{\bar{z}}, \mathbf{\bar{x}}) \) is a Lebesgue integrable function on \( I_{d_i} \).

If \( u(\mathbf{z}) \) is the solution of a linear PDE, then the kernels \( \{\Psi_i\} \) are linear functionals of the Green’s function for the PDE. See Butkovskiy [4] for examples of this type of representation of the solution operator of a PDE. We will henceforth refer to \( u \) as the solution function.

Numerical approximation assumptions. In order to numerically approximate \( u(\mathbf{z}) \), we first replace the possibly infinite dimensional problem with a finite dimensional problem. The dimensionality of the problem is reduced by introducing error in the following sense:

- Only a finite amount of information about the solution function is calculated. Any model of the solution based only on this information will merely approximate the true solution.

- Only a finite amount of information about the data functions is used to calculate the desired solution properties. We will refer to this as the data function sampling. Unless this information completely characterizes the data functions, the solution values that are calculated are also approximate.

Equation (2) is replaced by a relationship between the data function samples and the information about the solution function that is to be calculated.

For the rest of this paper, we restrict ourselves to problems where

(II) for each \( i \in \{1, \ldots, S\} \), values of the data function \( g_i \), sampled in its domain \( I_{d_i} \), are available to be used.
(III) values of the solution function at given locations in its domain \( \Omega \) are approximated, and
(IV) the error in approximating each solution value is bounded by a given value \( \epsilon \).

Thus, we are specifying what type of data can be used, what solution values to approximate, and how accurately these values are to be approximated. Note that we are not stipulating where the data functions can be sampled. Rather, we are stipulating that only pointwise evaluations of the functions can be used by any algorithm used to solve this problem.

Let \( Z \) represent the set of locations where the value of the solution function is needed, \( Z = \{ \hat{z}_j \mid j = 1, \ldots, N_a \} \). For brevity, we will use \( u_j \) to refer to \( u(\hat{z}_j) \), and we denote the set of solution values to be calculated by \( U \). We can use the following notation for any serial algorithm \( a \) with finite serial cost that solves a problem satisfying the above assumptions.

For each data function \( g_i \), the algorithm uses function values at some finite set of locations \( X_i = \{ \hat{z}_{i,k} \mid k = 1, \ldots, N_{a,i} \} \) in \( I_d \). And, for any particular solution value \( u_j \), the approximation to \( u_j \) calculated by algorithm \( a \) depends on values of \( g_i \) at some set of locations

\[
X_{i,j} = \{ \hat{z}_{i,j,k} \mid k = 1, \ldots, N_{a,i}(u_j) \} \subseteq X_i
\]

in \( I_d \). For brevity, we will use \( g_{i,j,k} \) to refer to \( g_i(\hat{z}_{i,j,k}) \). The total amount of data used to approximate \( u_j \) is \( N_a(u_j) = \sum_{i=1}^S N_{a,i}(u_j) \).

Data function assumptions. In order to identify whether a given algorithm solves a problem satisfying Assumptions (I)-(IV), we must be able to determine whether the error tolerance in approximating the individual solution values is satisfied (Assumption (IV)). The following assumption allows us to determine a priori lower bounds on the worst case error in an approximation.

Define \( C^m(R_k) \) to be the set of all functions that have continuous \( m \)th order partial derivatives on a set \( R_k \subset \mathbb{R}^k \). For \( g \in C^m(R_k) \), let \( \nabla_{k}^{(m)} g \) be a vector whose elements are some ordering of the \( m \)th order partial derivatives of \( g \) in \( \mathbb{R}^k \).

(V) For each \( i \in \{1, \ldots, S\} \), we assume that the data function \( g_i \) is known to be some member of a set \( G_i \) defined in the following way. \( G_i \) is the set of all functions \( g \) satisfying the properties

i) \( g(\bar{z}) \in C^{u_i}(I_d) \)

ii) \( \| \nabla_{d_i}^{(u_i)} g(\bar{z}) \|_{(1)} \leq T_i(\bar{z}) \) for all \( \bar{z} \in I_d \)

where \( u_i \) is a positive integer, \( \| \|_{(1)} \) is a vector norm, and \( T_i(\bar{z}) \) is a nonnegative function.

We also assume that any member of \( G_i \) is a permissible data function, and that any combination of data functions from the sets \( \{ G_i \} \) generate a possible solution to the PDE, with the following constraint. The inclusion of a given data function \( g_i \) in a set of data functions may force another data function \( g_j \), \( j \neq i \), to have given function and derivative values on the boundary of \( I_d \).

Assumption (V) states that the data functions are known to have a certain number of continuous derivatives, and that the magnitude of the largest partial derivatives is bounded by some known
function. This is the only information we are assuming about the data functions. This type of assumption on the data is similar to the assumptions that are made on the data and solution functions when specifying a priori bounds on the error introduced when using finite difference and finite element discretizations of a PDE to approximate the solution. The particular form of these assumptions is similar to that used by Traub and Woźniakowski [22].

3. Algorithm-dependent results

In this section, we describe the effect of the multiprocessor radius on the performance of parallel algorithms as a function of simple information-theoretic algorithm-dependent parameters. We implicitly assume that all algorithms discussed in this section solve problems that satisfy Assumptions (I)-(V), but only in the sense that we use the notation introduced in the previous section and use examples that do solve such problems.

3.1. Information-theoretic lower bounds

Lemma 3.1. The serial cost of an algorithm \( a \) is bounded from below by

\[
f_{(+)} \cdot (N_a(u_j) - 1)
\]

for any \( u_j \in U \).

Proof. For each solution value \( u_j \in U \), computing \( u_j \) requires that the data be reduced by a sequence of binary operations, and binary addition is assumed to be the fastest binary floating point operator. Thus, the serial cost of calculating \( u_j \) can be no smaller than the minimum serial cost of summing its required data. The proof follows immediately by a simple counting argument. (See also Lemma 1 in Kuck [16, page 95].) \( \square \)

Consider a parallel implementation of an algorithm \( a \) on a multiprocessor satisfying the assumptions of §2.1. A processor collaborates in the computation of a solution value \( u_j \in U \) if changing the results of the floating point operations calculated by that processor can change the value of \( u_j \). Define \( p_a(u_j) \) to be the number of processors that collaborate in the calculation of \( u_j \) for a given parallel implementation of \( a \).

Lemma 3.2. The computation cost of a parallel implementation of an algorithm \( a \) is bounded from below by

\[
f_{(+)} \cdot \max \left\{ \left\lceil \frac{N_a(u_j) - 1}{p_a(u_j)} \right\rceil, \left\lceil \log_2 N_a(u_j) \right\rceil \right\}
\]

for any \( u_j \in U \).

Proof. Since computing \( u_j \) requires that the data be reduced by a sequence of binary operations, the computation cost of calculating \( u_j \) on a multiprocessor can be no smaller than the minimum computation cost of summing its required data on the same multiprocessor. Since addition is a binary operator, \( N \) existing summands are replaced by at most \( N/2 \) results during any time interval of length \( f_{(+)} \). These results and any unused summands are the operands for the next step of the calculation. Thus, one lower bound on the computation
cost is \( f(+) \cdot \lfloor \log_2 N_a(u_j) \rfloor \). The other expression in the lemma comes immediately from (1), Lemma 3.1, and the definition of \( p_a(u_j) \). (See also Lemma 1 in Kuck [16, page 95]). \( \square \)

As in §2.1, let \( r(p_a(u_j)) \) be the radius of a \( p_a(u_j) \)-processor subset that has the minimum radius.

**Lemma 3.3.** The communication cost of a parallel implementation of an algorithm \( \textbf{a} \) is bounded from below by \( r(p_a(u_j)) \) for any \( u_j \in U \).

**Proof.** The lemma is a consequence of the fact that information is needed from all \( p_a(u_j) \) processors in order to calculate \( u_j \). Denote the subset of processors that collaborate in the calculation of \( u_j \) by \( V' \). Assume that a processor \( v \in V' \) has been designated to compute \( u_j \). Each processor in \( V' \) produces a result that is crucial to the calculation. Therefore, whether a result travels directly to \( v \) or is used by another processor to produce a new partial result that is then sent on, the time spent communicating is never less than the distance to \( v \) from the originating processor. Therefore, the maximum distance from the processors in \( V' \) to \( v \) is a lower bound on the communication cost. By definition, this is bounded from below by \( r(p_a(u_j)) \). \( \square \)

Define \( N_{a,*} \) to be the maximum amount of data required by an algorithm \( \textbf{a} \) to compute a single solution value,

\[
N_{a,*} = \max_{u_j \in U} N_a(u_j).
\]

**Theorem 3.4.** For a \( P \)-processor multiprocessor, a lower bound on the parallel cost of any parallel implementation of an algorithm \( \textbf{a} \) is

\[
\min_{p \in \{1, \ldots, P\}} \max \left\{ r(p), f(+) \cdot \left[ \frac{N_{a,*} - 1}{p} \right], f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor \right\}.
\]

**Proof.** The parallel cost is bounded from below by \( \max\{C_b, W_a\} \). Thus, by Lemmas 3.2 and 3.3,

\[
T_a \geq \max \left\{ r(p_a(u_j)), f(+) \cdot \left[ \frac{N_a(u_j) - 1}{p_a(u_j)} \right], f(+) \cdot \lfloor \log_2 N_a(u_j) \rfloor \right\}
\]

for each \( u_j \in U \). Since \( N_{a,*} = N_a(u_j) \) for some \( u_j \in U \), and since \( p_a(u_j) \) is a member of the set \( \{1, \ldots, P\} \) for any parallel implementation of algorithm \( \textbf{a} \) on this multiprocessor, the proof follows immediately. \( \square \)

Of the three terms in (3), \( r(p) \) is monotonically increasing, \( f(+) \cdot \lfloor (N_{a,*} - 1)/p \rfloor \) is monotonically decreasing, and \( f(+) \cdot \lfloor \log_2 N_a(u_j) \rfloor \) is constant (as a function of \( p \)). Thus, if \( P \geq N_{a,*} \), then either (3) achieves its minimum when \( r(p) \approx f(+) \cdot \lfloor (N_{a,*} - 1)/p \rfloor \) or \( r(p) \) is always less than \( f(+) \cdot \lfloor \log_2 N_a(u_j) \rfloor \) for \( p \in \{1, \ldots, P\} \), in which case the minimum is \( f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor \).

On an idealized multiprocessor where \( r(p) = 0 \) for all \( p \), the lower bound on the parallel cost described in Theorem 3.4 is \( f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor \) for large numbers of processors. If \( P \geq N_{a,*} \) and \( r(p) \leq \beta \cdot \lfloor \log_2 p \rfloor \) for some finite constant \( \beta \), then it follows immediately that (3) is bounded from above by

\[
\max\{f(+) \cdot \beta \cdot \lfloor \log_2 N_{a,*} \rfloor \}.
\]
Since (3) is always at least as large as \( f(+) \cdot \lceil \log_2 N_{a,*} \rceil \), this implies that (3) is also proportional to \( \lceil \log_2 N_{a,*} \rceil \) when \( r(p) \leq \beta \cdot \lceil \log_2 p \rceil \) and \( P \) is large. For example, this type of bound on \( r(p) \) holds when the multiprocessor is a clique or a hypercube. But, as the following theorem shows, the lower bound described by (3) grows much faster (as a function of \( N_{a,*} \)) when \( r(p) \) is a positive power of \( p \).

**Theorem 3.5.** For a given multiprocessor, assume that \( r(p) \geq \beta \cdot p^\gamma - \mu \) for positive constants \( \beta, \gamma, \) and \( \mu \). Then, independent of the number of processors, a lower bound on the parallel cost of any parallel implementation of an algorithm \( a \) on this multiprocessor is

\[
\max \left\{ \left( \beta \cdot \left( \frac{f(+)}{2 \cdot \beta} \cdot N_{a,*} \right)^{\gamma/(\gamma+1)} \right) - (\beta + \mu), f(+) \cdot \lceil \log_2 N_{a,*} \rceil \right\}.
\]

**Proof.** By Theorem 3.4, a lower bound on the parallel cost is

\[
\min_{0 < p < \infty} \max \left\{ r(p), f(+) \cdot \left[ \frac{N_{a,*} - 1}{p} \right], \left( f(+) \cdot \lceil \log_2 N_{a,*} \rceil \right) \right\} \tag{4}
\]

for any finite number of processors. By assumption, \( r(p) \geq \beta \cdot p^\gamma - \mu \) for positive constants \( \beta, \gamma, \) and \( \mu \). Redefine \( r(p) \) to be equal to \( \beta \cdot p^\gamma - \mu \). Then (4) is still a lower bound on the parallel cost since all we have done is use a worse lower bound on the communication cost.

Define \( LB(p) \) to be the function

\[
LB(p) = \max \left\{ r(p), f(+) \cdot \left[ \frac{N_{a,*} - 1}{p} \right], f(+) \cdot \lceil \log_2 N_{a,*} \rceil \right\},
\]

and let \( p_* \) denote some number that minimizes \( LB(p) \) over the domain \((0, \infty)\). (I.e. \( LB(p_*) \) has the same value as (4).) Next, consider some value \( p' \) for which

\[
r(p') \leq f(+) \cdot \left[ \frac{N_{a,*} - 1}{p'} \right]. \tag{5}
\]

That is, \( p' \) satisfies

\[
\beta \cdot (p')^\gamma - \mu \leq f(+) \cdot \left[ \frac{N_{a,*} - 1}{p'} \right]. \tag{6}
\]

By modifying the left hand side of (6) in such a way that its value is never decreased, and by modifying the right hand side in such a way that its value is never increased, we can replace (6) by the following simpler inequality that also guarantees that \( p' \) satisfies (5):

\[
(p')^{\gamma+1} \leq f(+) \cdot \frac{N_{a,*}}{2 \cdot \beta}.
\]

In particular, if we let

\[
p' = \left( f(+) \cdot \frac{N_{a,*}}{2 \cdot \beta} \right)^{1/(\gamma+1)},
\]
then $p'$ satisfies (5). Since $r(p)$ increases monotonically and since $f_{(+)} \cdot [(N_{a,*} - 1)/p]$ decreases monotonically, it follows that $LB(p)$ is minimized by some $p_*$ greater than or equal to $p'$. In particular, $r(p') \leq r(p_*)$. Since $f_{(+)} \cdot \lfloor \log_2 N_{a,*} \rfloor$ is always a lower bound on the parallel cost, this implies that

$$LB(p_*) \geq \max \{r(p'), f_{(+)} \cdot \lfloor \log_2 N_{a,*} \rfloor \}.$$

In consequence, a lower bound on the parallel cost is

$$\max \left\{ \left( \beta \cdot \left( \frac{f_{(+)} \cdot N_{a,*}}{2 \cdot \beta} \right)^{1/(y+1)} \right)^{1/3} - (\beta + \mu), f_{(+)} \cdot \lfloor \log_2 N_{a,*} \rfloor \right\}.$$

This proves the theorem. \(\square\)

**Corollary 3.6.** For any finite number of processors, a lower bound on the parallel cost of any parallel implementation of an algorithm $a$ is

$$\max \left\{ \frac{k \cdot t}{2} \cdot \left( \frac{f_{(+)} \cdot N_{a,*}}{k \cdot t} \right)^{1/(k+1)} - \left( \frac{k \cdot t}{2} \cdot \log_2 N_{a,*} \right) \right\},$$

if the multiprocessor is a $k$-dimensional array.

**Proof.** For a $k$-dimensional array, $r(p) \geq k \cdot t \cdot (p^{1/k} / 2 - 1)$. The corollary follows immediately from Theorem 3.5. \(\square\)

Thus, for a $k$-dimensional array multiprocessor, the communication cost, as represented by $r(p)$, determines the fundamental behavior of the lower bound in Theorem 3.4 for large $P$. In §3.2, we describe an example where $r(p)$ is the dominant factor in determining the parallel cost of a parallel algorithm, not just in determining a lower bound on the parallel cost.

### 3.2. Example communication bound algorithms

The following example problem shows that the intrinsic cost represented by the multiprocessor radius function can limit the performance of (good) parallel implementations on a $k$-dimensional array architecture. In particular, we describe conditions for which a given parallel algorithm $a$ is communication bound. We also describe conditions for which the corresponding serial algorithm $a$ is communication bound on a $k$-dimensional array architecture.

Consider the problem of calculating a single-valued solution function $u$ that is defined by

$$u = \int_0^1 g(x) \, dx.$$

An algorithm $a$ to approximate $u$ that uses a quadrature rule to calculate the integral has the form

$$\bar{u} = \sum_{k=1}^{N_{a,*}} \alpha_k g(x_k).$$

This algorithm contains $N_{a,*}$ multiplications and $N_{a,*} - 1$ additions.

Consider implementing this algorithm on $p$ processors of a $P$-processor $k$-dimensional array.
multiprocessor, where \( p \leq N_{a,*} \leq P \). A near-optimal parallel implementation with respect to the computation cost is the following. First, partition the data \( \{g(z_k)\} \) into \( p \) subsets each of which contains no more than \( \lceil N_{a,*}/p \rceil \) elements and allocate a different processor to each subset. Next, in parallel, use the processor allocated to a given subset to calculate the portion of the weighted sum in (7) corresponding to the subset. Finally, when all processors are finished, sum the remaining \( p \) partial results using a parallel fan-in algorithm [16]. The computation cost of this parallel implementation on a homogeneous multiprocessor is

\[
C_a = \left\lfloor \frac{N_{a,*}}{p} \right\rfloor \cdot (f(+)+f(\ast)) + (\lfloor \log_2 p \rfloor - 1) \cdot f(+) .
\]  

Since \( p \) processors are collaborating in such a parallel implementation, a lower bound on the communication cost is \( r(p) \). Assume that \( p = N_{a,*} \). Then (8) is minimized, but the communication cost is at least \( r(N_{a,*}) \). If \( r(N_{a,*}) > C_a \), then this parallel algorithm is communication bound since the communication cost is larger than the computation cost. The multiprocessor is a \( k \)-dimensional array, so this inequality is

\[
k \cdot t \cdot \left( \frac{N_{a,*}}{2} - 1 \right) > f(\ast) + f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor
\]
or

\[
N_{a,*}^k > \frac{2 \cdot f(\ast)}{k \cdot t} + \frac{2 \cdot f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor}{k \cdot t} .
\]

If we knew that

\[
\frac{2 \cdot f(\ast)}{k \cdot t} \geq \frac{2 \cdot f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor}{k \cdot t} ,
\]

then a sufficient condition for the parallel algorithm to be communication bound would be

\[
N_{a,*}^k > \frac{4 \cdot f(\ast)}{k \cdot t} .
\]

Using an argument similar to this, and taking into account all possible cases, it is straightforward to show that this parallel algorithm will be communication bound if

\[
N_{a,*} > \max \left\{ 16^k, 16^k \cdot \left( \frac{f(+)}{t} \right)^{2k}, 4^k \cdot \left( \frac{f(\ast) + f(+) + 1}{kt} \right)^k \right\} .
\]  

For example, if \( f(\ast) = f(+) = t \), then this bound is \( N_{a,*} > 16^k \). (This is an overly conservative bound. For example, if \( f(\ast) = f(+) = t \) and \( k \geq 2 \), then the radius forces this algorithm to be communication bound when \( N_{a,*} > \gamma^k \).) A similar analysis can be used to show that any parallel implementation for which \( p \propto N_{a,*}^\gamma \) where \( \gamma > k/\ell + 1 \) will be communication bound if \( N_{a,*} \) (and \( P \)) is large enough.

We can also show that the serial algorithm \( a \) is communication bound on a \( k \)-dimensional array multiprocessor if \( N_{a,*} \) is large enough and \( P = N_{a,*} \). First, let \( \mathcal{M} \) be the given \( k \)-dimensional array multiprocessor, and let \( \mathcal{M}' \) be an idealized multiprocessor constructed from \( \mathcal{M} \) in the fashion described in §2.2. As in §2.2, let \( \mathcal{M}'_{\ast} \) be an optimal parallel implementation of \( a \) on \( \mathcal{M}' \). Then expression (8) with \( p = N_{a,*} \) is an upper bound on \( T_{a*} \) since it represents the
execution time of a particular parallel implementation of a on M'.

Let \( \tilde{a} \) be an optimal parallel implementation of a on M. By Corollary 3.6,

\[
T_{\tilde{a}} \geq \max \left\{ \frac{k \cdot t}{2} \cdot \left( \left( \frac{f(+) \cdot N_{a,*}}{k \cdot t} \right)^{1/(k+1)} - 3 \right), f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor \right\}.
\]

Thus, \( T_{\tilde{a}} \) is larger than \( 2 \cdot T_{d_{a}} \) when

\[
\frac{k \cdot t}{2} \cdot \left( \left( \frac{f(+) \cdot N_{a,*}}{k \cdot t} \right)^{1/(k+1)} - 3 \right) > 2 \cdot (f(+) \cdot \lfloor \log_2 N_{a,*} \rfloor)
\]

or

\[
N_{a,*}^{1/(k+1)} > \left( \frac{k \cdot t}{f(+) \cdot k \cdot t} \right)^{1/(k+1)} \cdot \left( \frac{4 \cdot f(+) \cdot k \cdot t}{k \cdot t} + 3 + \frac{4}{k \cdot t} \cdot \lfloor \log_2 N_{a,*} \rfloor \right).
\]

Using a case-by-case analysis similar to that used to derive (9), it is straightforward to show that a sufficient condition for the serial algorithm a to be communication bound is that

\[
N_{a,*} > \max \left\{ (16)^{k+1} \cdot (8k + 8)^{2k+2} \cdot \left( \frac{f(+) \cdot k \cdot t}{kt} \right)^{2k} \cdot \left( \frac{8 \cdot (f(+) + f(+) \cdot k \cdot t)}{kt} + 6 \right)^{k+1} \cdot \frac{kt}{f(+) \cdot k \cdot t} \right\}.
\]

If \( f(+) = f(+) = t \), then the bound is \( N_{a,*} > 64^{k+1} \cdot (k + 1)^{2k+2} / k^{2k} \). (This is also an overly conservative bound. For example, if \( f(+) = f(+) = t \), then the radius forces this serial algorithm to be communication bound when \( N_{a,*} > 56^2 \) for \( k = 1 \), when \( N_{a,*} > 51^3 \) for \( k = 2 \), and when \( N_{a,*} > 44^4 \) for \( k = 3 \).) Similarly, if \( P \propto N_{a,*}^2 \) for some \( \gamma > k/k + 1 \), then \( T_{d_{a}} \geq 2 \cdot T_{d_{a}} \) if \( N_{a,*} \) is large enough. Thus, the radius function of a k-dimensional array strongly limits the attainable performance of algorithms for this example problem for large \( N_{a,*} \) (and \( P \)).

4. Problem-dependent results

By Theorem 3.5, if \( r(p) \geq \beta \cdot p^\gamma - \mu \) for positive constants \( \beta \), \( \gamma \), and \( \mu \) and if the number of processors is large, then the communication cost of a parallel algorithm determines the behavior of the lower bound on the parallel cost described in Theorem 3.4. In §3.2, by restricting ourselves to a specific algorithm, we were able to show the much stronger result that this condition on \( r(p) \) can force both the serial algorithm and good parallel implementations to be communication bound. In this section we show that similar results can be established for a given problem satisfying Assumptions (I)-(V) in §2.3.

4.1. Communication bound problems

Previously, we defined what it means for a parallel algorithm to be communication bound, and what it means for a serial algorithm to be communication bound on a given multiprocessor. In this section, we define what it means for a problem to be communication bound. Let \( q \) be a problem satisfying Assumptions (I)-(V) in §2.3. Let \( A(q) \) be the set of all serial algorithms that solve problem \( q \). Let \( M \) be a given multiprocessor, and let \( M' \) be the idealized multiprocessor described in §2.2.
Let $\bar{a}_q$ be the parallel algorithm that minimizes the parallel execution time over all parallel implementations on $M$ of all algorithms $a$ in $A(q)$. We call this the optimal parallel algorithm for $q$ on $M$. Let $\bar{a}'_q$ be the parallel algorithm that minimizes the parallel execution time over all parallel implementations on $M'$ of all algorithms $a$ in $A(q)$. We call this the optimal parallel algorithm for $q$ on $M'$. We say that $q$ is communication bound if $T_{a'} < T_{a'}/2$. Otherwise, we say that $p$ is computation bound. As for the case of serial algorithms, if a problem is communication bound, then the interprocessor communication subsystem is the dominant constraint on the ability of $M$ to solve the problem. If a problem is computation bound, then the intrinsic computational requirements of the problem and the speed of floating point computation are the dominant constraints.

4.2. Problem-dependent bounds

The argument used in §3.2 to show that a given serial algorithm is communication bound can also be used to show that a problem $q$ is communication bound on a multiprocessor $M$. We simply need to establish a lower bound on $N_{q,*}$ for all algorithms in $A(q)$, and an upper bound on the computational cost for a specific algorithm $a \in A(q)$. The lower bound on $N_{q,*}$, in conjunction with Theorem 3.4 and Theorem 3.5, can be used to calculate a lower bound on $T_{a'}$, the parallel cost of the optimal parallel algorithm for $q$ on $M$. The upper bound on the computation cost for a specific algorithm is also an upper bound on on $T_{a'}$, the parallel cost of the optimal parallel algorithm for $q$ on $M'$. The first two theorems in this section, proven in [23] and [24], allow us to calculate this information.

Lower bound on $N_{q,*}$. Let $\bar{B}(\bar{z}; \delta)$ denote a closed ball in $\mathbb{R}^{d_i}$ centered on $\bar{z}$ with a radius of $\delta$, $\delta \geq 0$. If $d_i = 0$, then we define $\delta$ to be zero. Define $N_{q,i}(u_j)$ to be the minimum number of data sampling locations in $I_{d_i}$ used by algorithms in $A(q)$ when approximating $u_j$.

**Theorem 4.1.** Let $q$ be a problem satisfying Assumptions (I)-(V). Let $\epsilon$ be the error bound on the approximation to the solution values specified by the problem. Let $\bar{z}_j$ be an element of $Z$, and let $u_j$ be the corresponding element of $U$. Let $i$ be an element of $\{1, \ldots, S\}$. If there exists a closed ball $\bar{B}(\bar{z}; \delta) \subset I_{d_i}$ of positive radius $\delta$ on which $\bar{\Psi}_i(\bar{z}_j, \bar{z})$ is either strictly positive or strictly negative and on which both $\bar{\Psi}_i(\bar{z}_j, \bar{z})$ and $\bar{T}_i(\bar{z})$ are bounded away from zero, then there exists a positive constant $c_{i,j}$ independent of $\epsilon$ such that

$$N_{p,i}(u_j) \geq \left\lceil c_{i,j} \cdot \epsilon^{-\frac{1}{\delta}} \right\rceil.$$  

The proof of this theorem is based on how different two solution functions can be when their data functions are identical at all of the sampling locations. See [23] or [24] for the details. The assumptions of the theorem are satisfied by most linear PDEs arising in scientific computing, and are used merely to ensure that $u_j$ cannot be calculated exactly with a finite amount of data from $I_{d_i}$. Since $N_{p,*} \geq N_{a}(u_j) \geq N_{a,i}(u_j) \geq N_{q,i}(u_j)$ for all algorithms $a$ in $A(q)$, this establishes the desired lower bound.
Corollary 4.2. If the conditions of Theorem 4.1 are satisfied for some \( \tilde{z}_j \in \mathbb{Z} \) and some \( i \in \{1, \ldots, S\} \), then

\[
T_{\tilde{a}} \geq \min_{p \in \{1, \ldots, P\}} \max \left\{ r(p), f_{(+)} \cdot \left[ c_{i,j} \cdot e^{-d_i/v_i} - 2 \right], f_{(+)} \cdot \left[ \log_2 \left( c_{i,j} \cdot e^{-d_i/v_i} - 1 \right) \right] \right\}.
\]

Here \( P \) is the number of processors in \( M \) and \( c_{i,j} \) is the constant from Theorem 4.1. In particular, if \( r(p) \geq \beta \cdot p^7 - \mu \) for positive constants \( \beta, \gamma, \) and \( \mu \), then

\[
T_{\tilde{a}} \geq \max \left\{ \beta \cdot \left( \frac{f_{(+)}(\gamma)}{2 \cdot \beta \cdot (c_{i,j} \cdot e^{-d_i/v_i} - 1)} \right)^{\gamma/(\gamma+1)} - (\beta + \mu), f_{(+)} \cdot \left[ \log_2 \left( c_{i,j} \cdot e^{-d_i/v_i} - 1 \right) \right] \right\}
\]

for any finite \( P \).

Proof. Since \( N_{a,*} \geq N_{q,i}(u_j) \) for any \( a \in A(q) \), the proof follows directly from Theorem 3.4, Theorem 3.5, and Theorem 4.1. \( \square \)

Upper bound on minimum computation cost. The next theorem describes an upper bound on \( N_a(u_j) \) for a particular serial algorithm \( a \). We will use this to derive an upper bound on the minimum computation cost over all parallel implementations of algorithms in \( A(q) \).

Consider the following algorithm for approximating a given \( u_j \in U \). For each \( i \in \{1, \ldots, S\} \) such that \( d_i = 0 \), compute

\[
\int_{L_{4,i}} \Psi_i(\tilde{z}_j, \tilde{x})g_i(\tilde{x}) \, d\tilde{x} = \Psi_i(\tilde{z}_j) \cdot g_i
\]

exactly. Call this value \( \bar{u}_{i,j} \). For each \( i \) such that \( d_i > 0 \), approximate \( \int_{L_{4,i}} \Psi_i(\tilde{z}_j, \tilde{x})g_i(\tilde{x}) \, d\tilde{x} \) in the following way:

a) Divide \( L_{4,i} \) into cubes of the form \( C_i(\tilde{x}_i,j; 2^{-v_i,j}) \), for some positive integer \( v_i,j \), where \( C_i(\tilde{x}; \delta) \) is the open \( d_i \)-dimensional cube centered on \( \tilde{x} \) with volume \( \delta^{d_i} \). Here \( l \) is the index for this set of cubes, \( l \in \{1, \ldots, 2^{v_i,j} \} \).

b) In each subcube \( C_i(\tilde{x}_i,j; 2^{-v_i,j}) \), consider \( v_i^{d_i} \) data sampling locations arranged on a uniform \( d_i \)-dimensional mesh with mesh spacing \( h = 2^{-v_i,j}/(v_i + 1) \). Let \( \tilde{g}_{i,j,i}(\tilde{x}) \) be the unique polynomial of degree at most \( v_i - 1 \) in each variable that interpolates \( g_i(\tilde{x}) \) at the indicated sampling locations.

c) Approximate \( \int_{L_{4,i}} \Psi_i(\tilde{z}_j, \tilde{x})g_i(\tilde{x}) \, d\tilde{x} \) by

\[
\bar{u}_{i,j} \equiv \sum_{l=1}^{2^{v_i,j}} \int_{C_i(\tilde{x}_i,j; 2^{-v_i,j})} \Psi_i(\tilde{z}_j, \tilde{x})\tilde{g}_{i,j,i}(\tilde{x}) \, d\tilde{x}.
\]

Finally, approximate \( u_j \) by the following expression,

\[
\bar{u}_j = \sum_{i=1}^{S} \bar{u}_{i,j}.
\]
We get an algorithm for any problem \( q \) by using this technique for each \( u_j \in U \) and using enough data function samples to satisfy the error bound. We refer to this type of algorithm as a Green's function method since it uses the Green's function directly in the calculation.

**Theorem 4.3.** Let \( q \) be a problem satisfying Assumptions (I)-(V). Let \( \epsilon \) be the error bound on the approximation to the solution values specified by the problem. For each \( u_j \in U \), there exists a set of finite positive constants \( \{ \hat{c}_{i,j} \} \), each of which is independent of \( \epsilon \), with the following property. There exists a Green's function method \( a \) in \( A(q) \) such that for each \( u_j \in U \) and \( i \in \{1, \cdots, S\} \)

\[
N_{a,i}(u_j) \leq \hat{c}_{i,j} \cdot \epsilon^{-\frac{2}{\epsilon^2}} + 1.
\]

The proof of the theorem is based on bounding the error introduced by using the interpolating polynomials \( \{\hat{g}_{i,j,l} \} \) instead of the true data functions in the Green's function method. See [23] or [24] for the details. Since \( N_{a,s} = \max_{u_j \in U} \sum_{i=1}^{S} N_{a,i}(u_j) \), this theorem describes an upper bound on the size of \( N_{a,s} \) for the specified algorithm.

The next theorem describes an upper bound on \( T_{a,q}^* \). It is derived from an upper bound on the computation cost of a particular parallel implementation of a Green's function method. Note that the expression \( [P/|U|] \) is used in the first inequality in the theorem, where \( |U| \) is the number of solution values to be calculated. This arises because one of the two parallel implementations used to establish the result calculates each solution value on a distinct subset of processors, and the expression \( [P/|U|] \) is an upper bound on the number of processors in the smallest subset. Since this approach only makes sense if \( |U| \leq P \), a different parallel implementation is used to calculate a bound when \( |U| > P \).

**Theorem 4.4.** Let \( q \) be a problem satisfying Assumptions (I)-(V). Let \( \epsilon \) be the error bound on the approximation to the solution values specified by the problem. Let \( |U| \) be the number of solution values in \( U \). Let \( M \) be a homogeneous \( P \)-processor multiprocessor, and let \( M' \) be the idealized multiprocessor constructed from \( M \) in the fashion described in §2.2. Let \( \{ \hat{c}_{i,j} \} \) be the constants from Theorem 4.3. Then

\[
T_{a,q}^* \leq \frac{\max_{j \in \{1, \cdots, |U|\}} \min_{p \in \{1, \cdots, [P/|U|]\}} \left( \sum_{i=1}^{S} \left( \hat{c}_{i,j} \cdot \epsilon^{-\frac{2}{\epsilon^2}} + 1 \right) \frac{(f_{(+)} + f_{(*)})}{p} \right)}{f_{(+)} + ([\log_p P] - 1) f_{(+)}}
\]

if \( P \geq |U| \), and

\[
T_{a,q}^* \leq \frac{|U|}{P} \cdot \max_{j \in \{1, \cdots, |U|\}} \left( \sum_{i=1}^{S} \left( \hat{c}_{i,j} \cdot \epsilon^{-\frac{2}{\epsilon^2}} + 1 \right) \cdot (f_{(+)} + f_{(*)}) - f_{(+)} \right)
\]

if \( P < |U| \).

**Proof.** For fixed \( i, j, \) and \( l \), let the set \( \{ \hat{x}_{i,j,l,k} | k = 1, \cdots, v_{i,l}^l \} \) denote the data sampling locations in \( C_l(\hat{x}_{i,j,l}; 2^{-v_{i,l}}) \) used by a Green's function method \( a \). Then, in this subcube, the
polynomial interpolant $\tilde{g}_{i,j,l}$ can be represented by

$$
\sum_{k=1}^{v_{i,j,l}} g_{i,j,l}(\tilde{x}_{i,j,l,k}) \cdot \Lambda_{i,j,l,k}(\tilde{x})
$$

where $\{\Lambda_{i,j,l,k}(\tilde{x})\}$ are Lagrange basis functions for these sampling locations. (See Prenter [18, pages 118-127] or Worley [23, pages 131-134].)

Using this notation, the expression for $\tilde{u}_{i,j}$ becomes

$$
\tilde{u}_{i,j} = \sum_{j=1}^{2^{v_{i,j,l}}} \int_{C_i(x_{i,j,l})} \Psi_i(\tilde{x}_j, \tilde{x}) \cdot \left( \sum_{k=1}^{v_{i,j,l}} g_l(\tilde{x}_{i,j,l,k}) \cdot \Lambda_{i,j,l,k}(\tilde{x}) \right) d\tilde{x}
$$

Each factor $\int_{C_i(x_{i,j,l})} \Psi_i(\tilde{x}_j, \tilde{x}) \cdot \Lambda_{i,j,l,k}(\tilde{x}) d\tilde{x}$ is independent of the data, and can be precomputed. Call it $r_{i,j,l,k}$. The expression for $\tilde{u}_{i,j}$ is then

$$
\tilde{u}_{i,j} = \sum_{j=1}^{2^{v_{i,j,l}}} \sum_{k=1}^{v_{i,j,l}} r_{i,j,l,k} \cdot g_l(\tilde{x}_{i,j,l,k})
$$

and $N_a(u_j) = v_l^{d_l} \cdot 2^{v_{i,j,l}}$. Similarly, the expression for the approximation to $u(\tilde{x}_j)$ is now

$$
\tilde{u}_j = \sum_{j=1}^{S} \sum_{l=1}^{2^{v_{i,j,l}}} \sum_{k=1}^{v_{i,j,l}} r_{i,j,l,k} \cdot g_l(\tilde{x}_{i,j,l,k})
$$

and $N_a(u_j) = \sum_{i=1}^{S} v_l^{d_l} \cdot 2^{v_{i,j,l}}$.  

By (10), using algorithm a to approximate $u(\tilde{x}_j)$ requires $N_a(u_j)$ floating point multiplications and $N_a(u_j) - 1$ floating point additions. To implement this algorithm using $p$ processors of a $P$-processor multiprocessor, use the same technique described in §3.2: partition the data equally among the processors, calculate as much as possible in parallel, synchronize, and sum the remaining $\min\{N_a(u_j), p\}$ partial results using a parallel fan-in algorithm. Since computing $u_j$ in this fashion is independent of computing any other member of $U$, the same approach can be used for all other solution values, as long as the total number of processors used does not exceed $P$. Assume that $|U| \leq P$. Then we can allocate $[P/|U|]$ processors to the computation of each $u_j$, but use only as many processor as is efficient for each computation. The parallel cost of this parallel implementation on $M'$ is

$$
\max_{j \in \{1, \ldots, |U|\}} \min_{p \in \{1, \ldots, [P/|U|]\}} \left( \frac{\sum_{i=1}^{S} N_a(u_j)}{p} \cdot (f(+) + f(\star)) + ([\log_2 p] - 1) \cdot f(+) \right).
$$

Next, assume that $|U| > P$. Then a different approach is required. Instead, partition the
set of solution values into $P$ subsets each of which contains no more than $\lceil|U|/P\rceil$ elements, and assign a distinct processor to the calculation of the solution values in each subset. The parallel cost of this parallel implementation on $M'$ is no more than

$$
\left\lceil\frac{|U|}{P}\right\rceil \cdot \max_{f \in \{f_{\pm}\}} \left( \left( \sum_{i=1}^{S} N_{a,i}(u_j) \right) \cdot (f_{\pm} + f_{\mp}) - f_{\pm} \right),
$$

which is an upper bound on the serial cost of calculating a single subset of the solution values.

If $a \in A(q)$, then (11) and (12) are upper bounds on $T_{M'}$, the parallel cost of the optimal parallel algorithm for $q$ on $M'$. Therefore, the statement of the theorem follows immediately from Theorem 4.3, which describes an upper bound on $N_{a,i}(u_j)$ for some Green's function method $a \in A(q)$. 

4.3. Sufficient conditions for a problem to be communication bound

The following theorem describes conditions under which a problem will be communication bound on a homogeneous multiprocessor whose radius function is a positive power of the number of processors. Thus, a necessary condition for a homogeneous multiprocessor to be "adequate" for such problems is that its radius grow more slowly than this. The theorem is stated in the following way: There exists a constant $\epsilon_*$ that is a function of the solution operator and the smoothness of the data functions such that the problem will be communication bound if the error tolerance $\epsilon$ is less than $\epsilon_*$ and if a large number of processors are available. Thus, to determine whether a problem is communication bound on a given multiprocessor, first calculate $\epsilon_*$, then compare $\epsilon$ with $\epsilon_*$. As long as $\epsilon_*$ is positive, the problem is guaranteed to be communication bound if $\epsilon$ is small enough and the number of processors is large enough.

**Theorem 4.5.** Let $q$ be a problem satisfying Assumptions (I)-(V). Let $\epsilon$ be the error bound on the approximation to the solution values specified by the problem. Let $\beta$, $\gamma$, and $\mu$ be positive constants, and assume that $r(p) \geq \beta \cdot p^\gamma - \mu$ for a given homogeneous $P$-processor multiprocessor $M$. If, for some $\tilde{z}_j \in Z$ and some $i \in \{1, \ldots, S\}$, there exists a closed ball $B(\tilde{z}_j; \delta) \subseteq I_{a,i}$ of positive radius $\delta$ on which $\Psi_i(\tilde{z}_j, \tilde{x})$ is either strictly positive or strictly negative and on which both $\Psi_i(\tilde{z}_j, \tilde{x})$ and $T_i(\tilde{x})$ are bounded away from zero, then there exists a positive constant $\epsilon_*$, independent of $\epsilon$, and a finite function $P_*(\epsilon)$ such that $q$ is communication bound on $M$ if $\epsilon \leq \epsilon_*$ and $P \geq P_*(\epsilon)$.

**Proof.** By Corollary 4.2, $T_{M}$ is bounded from below by

$$
T_{M} \geq \beta \cdot \left( \frac{f_{\pm}}{2 \cdot \beta} \cdot (c_{i,j} \cdot \epsilon^{-\frac{\Delta_i}{\gamma_i}} - 1) \right)^{\gamma/(\gamma+1)} - (\beta + \mu).
$$

Let

$$
P_*(\epsilon) = |U| \cdot \max_{u_j \in U} \sum_{i=1}^{S} \left( \tilde{c}_{i,j} \cdot \epsilon^{-\frac{\Delta_i}{\gamma_i}} + 1 \right),
$$
where the constants $\{\bar{e}_{i,j}\}$ are from Theorem 4.3. By Theorem 4.4, if $P \geq P_*(\epsilon)$, then

$$T_{a'_q} \leq \max_{j \in \{1,\ldots,[U/l]\}} \left( f(\ast) + f(+) \cdot \left( \sum_{i=1}^{S} \left( \bar{e}_{i,j} \cdot \epsilon^{-d_i/v_i} + 1 \right) \right) \right)$$

$$\leq \max_{j \in \{1,\ldots,[U/l]\}} \left( f(\ast) + f(+) \cdot \log_2 \left( \sum_{i=1}^{S} \left( \bar{e}_{i,j} \cdot \epsilon^{-d_i/v_i} + 1 \right) \right) \right)$$

$$\leq \max_{j \in \{1,\ldots,[U/l]\}} \left( f(\ast) + f(+) \cdot \max_{j \in \{1,\ldots,[U/l]\}} \max_{i \in \{1,\ldots,S\}} \log_2 \left( S \cdot \bar{e}_{i,j} \cdot \epsilon^{-d_i/v_i} + 1 \right) \right)$$

$$= \left( f(\ast) + f(+) \right) \cdot \max_{j \in \{1,\ldots,[U/l]\}} \max_{i \in \{1,\ldots,S\}} \log_2 \left( S \cdot \bar{e}_{i,j} \cdot \epsilon^{-d_i/v_i} + 1 \right) \quad (14)$$

Let

$$\epsilon' = \max_{j \in \{1,\ldots,[U/l]\}} \max_{i \in \{1,\ldots,S\}} \bar{e}_{i,j}^{d_i/v_i}.$$ 

If $\epsilon < \epsilon'$, then $\bar{e}_{i,j} \cdot \epsilon^{-d_i/v_i} \geq 1$ for some $i$ and the following inequality can be derived from (14):

$$T_{a'_q} \leq \left( f(\ast) + f(+) \right) + f(+) \cdot \max_{j \in \{1,\ldots,[U/l]\}} \max_{i \in \{1,\ldots,S\}} \log_2 \left( 2 \cdot \bar{e}_{i,j} \cdot \epsilon^{-d_i/v_i} \right)$$

$$\leq \left( f(\ast) + f(+) \right) + f(+) \cdot \left( \max_{j \in \{1,\ldots,[U/l]\}} \max_{i \in \{1,\ldots,S\}} \log_2(2 \cdot \bar{e}_{i,j}) + \max_{i \in \{1,\ldots,S\}} d_i \cdot \log_2\epsilon^{-1} \right) \quad (15)$$

Since the upper bound on $T_{a'_q}$ in (15) grows like $\log_2(1/\epsilon)$ as $\epsilon$ decreases and the lower bound on $T_{a_q}$ in (13) grows like $\epsilon^{-d_i/v_i} \cdot \log_2(e/(v_i \cdot (\epsilon + 1)))$ as $\epsilon$ decreases, there will be some $\epsilon''$ for which the upper bound in (15) is less than half the lower bound in (13). Let $\epsilon* = \min\{\epsilon', \epsilon''\}$. If $\epsilon \leq \epsilon*$ and $P \geq P_*(\epsilon)$, then $T_{a'_q} \leq T_{a_q}/2$. This proves the theorem. □

As mentioned earlier, the assumptions in Theorem 4.5 on the kernels $\{\Psi_i\}$ and the functions $\{Y_i\}$ will be satisfied by most linear PDEs arising in scientific computing, and they do not constitute a significant restriction on the application of this theorem.

Note that the restriction to homogeneous multiprocessors is not necessary. A similar result will hold for a heterogeneous multiprocessor if the ratio of $f_\ast$ on the slowest processor to $f(\ast)$ on the fastest processor is bounded. Also note that the conditions under which a problem will be communication bound can be significantly tightened if given more information about the problem. For example, for many problems there is a great deal of shared work possible when computing the solution values, and all processors contribute to the calculation of all solution values for good parallel algorithms. This can significantly decrease $P_*(\epsilon)$. Also, generally all $i \in \{1, \ldots, S\}$ will satisfy the problem assumptions, and this will increase $\epsilon*$. Both of these changes make it more likely that a problem will be communication bound. But, even without knowing more details, we are able to identify when a problem will necessarily be communication bound.
5. Scaling results

Scaling a multiprocessor architecture increases or decreases the number of processors in the multiprocessor while keeping certain attributes of the architecture fixed. In particular, we define scaling for the example architectures of §2.1 in the following way. The graph of the scaled multiprocessor architecture is of the same type as before scaling, and each type of component has the same parameters. For example, when scaling a square $k$-dimensional array multiprocessor with $P$ processors, the graph of the new multiprocessor will still be a square $k$-dimensional array. The processor and communication abilities will also be unchanged, but the number of processors will now be $Q$ for some $P \neq Q$. Thus, scaling defines a family of multiprocessors with similar architectures. We will refer to a particular multiprocessor as an instance of this architectural family and to the number of processors in an instance as its size. We will use the term scaling up to mean increasing the size of a multiprocessor architecture.

Scaling a problem alters the problem specifications in such a way that the serial complexity of the algorithms used to solve it changes. The problem parameters that are normally free to be varied are the solution values to be approximated and the error bound to be satisfied. Therefore, to scale the problem, one or both of these parameters must be changed. Scaling the problem defines a family of similar problems, all approximating the solution of the PDE. We will refer to a given set of specifications as a problem instance. The size of a problem instance is the minimum serial cost of algorithms that solve the problem instance. We will use the term scaling up to mean changing the problem specifications in such a way that its size increases.

We say that a multiprocessor architecture scales for a problem if the minimum parallel cost when solving the problem can be bounded independent of the problem size by scaling up the architecture. In previous work [23], [24], we showed that no multiprocessor architecture scales for our problems when given a reasonable assumption on how the size of a problem can grow. In this section, we briefly describe this result again, and then show how the radius function of a multiprocessor can exacerbate the increase in the minimum parallel cost.

5.1. Problem assumption VI

Commonly, increasing the size of a problem indicates that the number of solution values and the amount of data used increase, and that the error bound decreases. For some applications the solution is desired at only a fixed set of locations, and only the other two parameters will vary. But, in both cases, increasing the size of the problem results in a better approximation to the solution function. It is simple to increase the size of a problem without suffering an increased parallel cost if only the number of solution values is increased and if enough idle processors are available. But continually increasing the number of approximate solution values will not lead to a better solution unless the error in these approximate values also decreases. We do not consider it reasonable to increase the size of the problem unless there is some advantage gained by doing so. This motivates the following assumption on how the size of problem is increased.

(VI) For a given problem instance, denote the set of locations where the solution is to be approximated by $Z$, and denote the error bound on these approximations by $\epsilon$. If the size

\footnote{For example, see [17] for a discussion of parameterized architectures.}
of the problem grows and \( Z' \) and \( \epsilon' \) are the corresponding parameters of the new instance, then we assume that \( Z \subseteq Z' \) and

\[
\epsilon' \leq \epsilon \cdot (|Z|/|Z'|)^\alpha
\]

for some positive \( \alpha \) independent of the scaling. Here \( |Z| \) represents the number of locations in the set.

Note that this assumption is not on the problem, but rather on how we permit the problem to grow. By this assumption the problem size is allowed to grow only if the error bound also decreases. Moreover, if the problem size grows without bound, then the error bound goes to zero. Assumption (VI) is unnecessarily restrictive, but it is sufficient to establish the results in this section. In particular, the assumption that \( Z \subseteq Z' \) is made merely to keep the assumption simple. See Worley [23, pages 65-66] for a less restrictive assumption.

5.2. Problem-scaling bound

The following theorem shows that no architecture will scale for a linear PDE that satisfies Assumptions (I)-(VI) and the assumptions of the theorem.

**Theorem 5.1.** Let \( q \) be a problem instance satisfying Assumptions (I)-(V), and assume that all scalings of \( q \) satisfy Assumption (VI). Assume that, for some \( i \in \{1, \ldots, S\} \), there exists a \( \bar{z}_j \in Z \) with the following property: there exists a closed ball \( \bar{B}_i(\bar{z}, \delta) \subseteq I_d \) of positive radius \( \delta \) on which \( \Psi_i(\bar{z}_j, \bar{z}) \) is either strictly positive or strictly negative and on which both \( \Psi_i(\bar{z}_j, \bar{z}) \) and \( \Gamma_i(\bar{z}) \) are bounded away from zero. Also assume that \( f_i(+) \) is positive and bounded away from zero for all permissible multiprocessor architectures. Then, if the size of the problem increases without bound in a fashion consistent with Assumption (VI), so will the parallel cost, independent of the algorithm and of the number of processors used.

**Proof.** The proof follows immediately from Corollary 4.2 and the assumption on how problems are allowed to scale. As the size of a problem increases, the error tolerance decreases, and the logarithmic term in Corollary 4.2 increases. If the problem size increases unboundedly, then \( \epsilon \) goes to zero, and the logarithmic term increases unboundedly. See also [23] and [24].

5.3. Effect of \( r(p) \) on the asymptotic growth of the parallel cost

By Theorem 5.1, we usually can’t bound the parallel cost as the size of the problem increases. But a good multiprocessor architecture won’t exacerbate this increase. In this section we examine how the communication capabilities of a multiprocessor affect the parallel cost as the size of both the architecture and the problem increase. In particular, we show that most problems can become communication bound as the sizes increase if the multiprocessor radius function grows as a positive power of the number of processors. Thus, a necessary condition for a multiprocessor to be “adequate” in this context is that its radius grow more slowly than a positive power of \( p \).
Theorem 5.2. Let \( q \) be a problem instance satisfying Assumptions (I)-(V), and assume that all scalings of \( q \) satisfy Assumption (VI). Assume that there exists a \( z_j \in \mathbb{Z} \) and some \( i \in \{1, \ldots, S\} \) such that \( d_i \neq 0 \) and the following property holds: there exists a closed ball \( B_{\delta}(\bar{z}_i, \delta) \subset I_{\delta} \) of positive radius \( \delta \) on which \( \Psi_i(z_j, \bar{x}) \) is either strictly positive or strictly negative and on which both \( \Psi_i(z_j, \bar{x}) \) and \( T_i(\bar{x}) \) are bounded away from zero. Let \( \beta, \gamma, \) and \( \mu \) be positive constants, and assume that \( r(p) \geq \beta \cdot p^n - \mu \) for all scalings of a given homogeneous multiprocessor \( M \). Then the following conditions hold.

1) There exists a scaling of \( q \) and a multiprocessor instance such that the new problem instance is communication bound on this multiprocessor. Moreover, this problem instance remains communication bound for all scalings of the multiprocessor larger than the given instance.

2) Let \( R(p) \) be the ratio \( T_{a_0}/T_{a_1} \) on a multiprocessor instance of size \( p \). Then the limit \( \lim_{p \to \infty} R(p) \) grows unboundedly as a function of the problem size.

Proof. The first part of this theorem is essentially a restatement of Theorem 4.5. As a problem scales up in size, eventually \( \epsilon \) will be less than the constant \( \epsilon_* \) from Theorem 4.5. Once \( \epsilon \) is smaller than this threshold, the problem instance will be communication bound on any multiprocessor instance for which \( P \geq P_\epsilon(\epsilon) \).

From (13) and (15),

\[
R(p) \geq \frac{\beta \cdot \left( \frac{f_+}{2} \cdot (c_{i,j} \cdot \epsilon - \frac{d_i}{v_i} - 1) \right)^{\gamma/(\gamma+1)} - (\beta + \mu)}{(f_+ + f_{++}) + f_{+} \cdot \left( \max_{i \in \{1, \ldots, S\}} \max_{i \in \{1, \ldots, S\}} \frac{\log_2(2 \cdot S \cdot \epsilon_{i,j})}{\log_2 \epsilon^i} + \frac{d_i \cdot \log_2 \epsilon^{-1}}{v_i} \right)}
\]

if \( \epsilon \leq \epsilon_* \) and \( P \geq P_\epsilon(\epsilon) \). Thus, by Assumption (VI), if the size of the problem increases unboundedly, then \( \epsilon \) goes to zero, and \( \lim_{p \to \infty} R(p) \) grows unboundedly. This proves the theorem. \( \square \)

As mentioned earlier, the assumption that the multiprocessor is homogeneous is unnecessary. Generalizing Theorem 4.5 to take into account heterogeneous multiprocessors is all that is needed to generalize this result.

In summary, if \( r(p) \geq \beta \cdot p^n - \mu \) for positive constants \( \beta, \gamma, \) and \( \mu \), then the communication costs will eventually be the dominant constraint on the achievable parallel performance as the problem and multiprocessor sizes are increased. In particular, the following corollary is an immediate consequence.

Corollary 5.3. Assume that the graph of a multiprocessor is a \( k \)-dimensional array, and that the transmission time \( t \) is bounded away from zero. Assume that the PDE satisfies the conditions in Theorem 5.2. Then, for large problem sizes, the communication cost determines the minimum parallel cost as the multiprocessor is scaled up.

Ultimately, the communication capabilities of any multiprocessor is constrained by the three dimensionality of the physical world and by the speed of light. This motivates the next corollary.
Corollary 5.4. Assume that each processor in a multiprocessor is a cube with a fixed nonzero volume. Assume that the PDE satisfies the conditions in Theorem 5.2. Then, for large problem sizes, the communication cost determines the minimum parallel cost as the multiprocessor is scaled up.

Proof. Assume that the volume of a processor is $v$ in some standard unit. Then a set of $p$ processors will take up a volume of at least $pv$, and cover a region whose maximum width is at least $(pv)^{1/3}$. Any message between processors must travel from a surface of the sending processor to a surface of the receiving processor. Let the physical diameter of a set be the maximum distance between the nearest surfaces of two processors in the set. Then the physical diameter is greater than or equal to $(pv)^{1/3} - 2v^{1/3}$.

As in §2.1, let the center processor be the one that minimizes the maximum distance between itself and all of the others, where distance is now measured between closest surfaces. Let the physical radius be the maximum distance between the center and the other processors. Then half the physical diameter minus half the width of the center processor is a lower bound on the physical radius. In consequence,

$$\frac{(pv)^{1/3} - 3 \cdot v^{1/3}}{2}$$

is also a lower bound on the physical radius. Since no message can travel faster than the speed of light, the radius of this set of processors is at least

$$\frac{(pv)^{1/3} - 3 \cdot v^{1/3}}{2 \cdot c},$$

where $c$ is the speed of light in these units. If $\beta = v^{1/3}/(2 \cdot c)$, $\gamma = 1/3$, and $\mu = 3 \cdot v^{1/3}/(2 \cdot c)$, then $r(p) \geq \beta \cdot p^\gamma - \mu$. The result then follows from Theorem 5.2.  

Using current technologies, the speed of light is not the only restriction on transmission speed, and $r(p) \geq \beta \cdot p^\gamma - \mu$ for $\beta$ that is much larger than that calculated in this corollary. The important point of this corollary is that the communication cost will eventually determine the achievable performance for any physically-realizable multiprocessor.

6. Conclusions

The information-theoretic algorithm-dependent parameters introduced in §3 allowed us to calculate a lower bound on the parallel cost that included one aspect of the communication cost. By taking into account the intent of the algorithm, to approximate the solution of a continuous problem with a given structure, we were then able to calculate a similar lower bound on the parallel cost of optimal parallel algorithms on a given multiprocessor. We were also able to calculate an upper bound on the parallel cost of optimal parallel algorithms on an idealized multiprocessor for which the communication cost is always zero. Using these two results, we proved that the communication cost will be the dominant factor in determining the performance of optimal parallel algorithms for large problems as multiprocessors scale up in size. These results are a consequence of very general assumptions, and we expect the interprocessor communication subsystem to be the major constraint on the performance for problem and multiprocessor
sizes much smaller than those indicated by the theory described here. Whether this analysis indicates a practical limitation for a given problem must be examined on a case-by-case basis.

Acknowledgements

We thank Al Geist and Barry Peyton for their helpful suggestions on the presentation of the material in this paper.

References


### INTERNAL DISTRIBUTION

1. B. R. Appleton  
2. E. F. D'Azevedo  
3. J. J. Dongarra  
4. J. B. Drake  
5. T. H. Dunigan  
6. R. E. Flanery  
7. G. A. Geist  
8. L. Gray  
9-10. R. F. Harbison  
11. M. T. Heath  
12. E. R. Jessup  
13. M. R. Leuze  
14-18. F. C. Maienschein  
19. E. G. Ng  
20. C. E. Oliver  
21. G. Ostrouchov  
22. B. W. Peyton  
23. W. M. Post  
24-28. S. A. Raby  
29. C. H. Romine  
30-34. R. C. Ward  
35-39. F. H. Worley  
40. J. J. Dorning (EPMD Advisory Committee)  
41. R. M. Haralick (EPMD Advisory Committee)  
42. J. E. Leiss (EPMD Advisory Committee)  
43. M. F. Wheeler (EPMD Advisory Committee)  
44. N. Moray (EPMD Advisory Committee)  
45. Central Research Library  
46. ORNL Patent Office  
47. K-25 Plant Library  
48. Y-12 Technical Library  
49. Document Reference Station  
50-51. Laboratory Records - RC  
52. Dr. Loyce M. Adams, Department of Applied Mathematics, University of Washington, Seattle, WA 98195  
53. Dr. Christopher R. Anderson, Department of Mathematics, University of California, Los Angeles, CA 90024  
54. Dr. Robert G. Babb, Department of Computer Science and Engineering, Oregon Graduate Center, 19600 N.W. Walker Road, Beaverton, OR 97006  
55. Dr. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802  
56. Dr. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratory, Albuquerque, NM 87185  
57. Dr. Dov S. Bai, Department of Mathematics, Utah State University, Logan, UT 84322-4125  
58. Dr. David H. Bailey, NASA Ames, Mail Stop 258-5, NASA Ames Research Center, Moffett Field, CA 94035
59. Dr. Robert E. Benner, Parallel Processing Division, 1413, Sandia National Laboratories, Albuquerque, NM 87185

60. Dr. Marsha J. Berger, Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012

61. Prof. Ake Bjorck, Department of Mathematics, Linkoping University, Linkoping 58183, Sweden

62. Dr. John H. Bolstad, L-16, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

63. Dr. James C. Browne, Department of Computer Sciences, University of Texas, Austin, TX 78712

64. Dr. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

65. Dr. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109


67. Dr. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024

68. Dr. Jagdish Chandra, Army Research Office, P. O. Box 12211, Research Triangle Park, North Carolina 27709

69. Dr. Melvyn Ciment, National Science Foundation, 1800 G Street, NW, Washington, DC 20550

70. Prof. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853

71. Dr. Jane K. Cullum, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598

72. Dr. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720

73. Dr. George Cybenko, Center for Supercomputing Research & Development, 104 South Wright Street, Urbana, IL 61801-2932

74. Ms. Helen Davis, Computer Science Department, Stanford University, Stanford, CA 94305

75. Professor Larry Dowdy, Computer Science Department, Vanderbilt University, Nashville, TN 37235

76. Dr. Iain Duff, CSS Division, Harwell Laboratory, Didcot, Oxon OX11 ORA, ENGLAND
77. Dr. Stanley Eisenstat, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520

78. Dr. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742

79. Dr. Peter G. Eltgroth, L-298, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

80. Prof. David Fisher, Department of Mathematics, Harvey Mudd College, Claremont, CA 91711

81. Professor Geoffrey C. Fox, Physics Department, MS 356-48, California Institute of Technology, Pasadena, CA 91125

82. Dr. Chris Fraley, Department of Mathematics and Statistics, Utah State University, Logan, UT 84322-3900

83. Dr. Paul O. Frederickson, RIACS, NASA Ames Research Center, Moffet Field, CA 94035

84. Dr. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650

85. Professor Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47401

86. Dr. C. William Gear, Computer Science Department, University of Illinois, Urbana, IL 61801

87. Dr. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8

88. Dr. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

89. Dr. Gene Golub, Computer Science Department, Stanford University, Stanford, CA 94305

90. Prof. John L. Gustafson, Ames Laboratory, 236 Wilhelm Hall, Iowa State University, Ames, IA 50011-3020

91. Dr. Joseph F. Grcar, Division 8331, Sandia National Laboratories, Livermore, CA 94550

92. Dr. Eric Grosse, 2C 471, 600 Mountain Avenue, Murray Hill, NJ 07922

93. Dr. William D. Gropp, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

94. Dr. Gerald W. Hedstrom, L-71, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550
95. Dr. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P. O. Box 481, Houston, TX 77001
96. Dr. John L. Hennessy, CIS 208, Stanford University, Stanford, CA 94305
97. Dr. N. J. Higham, Department of Mathematics, University of Manchester, Gtr Manchester, M13 9PL, ENGLAND
98. Dr. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332
99. Dr. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550
100. Dr. Ilse Ipsen, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520
101. Dr. Lennart S. Johnsson, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520
102. Dr. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309
103. Dr. Bo Kågström, Institute of Information Processing, University of Umeå, 5-901 87 Umeå, Sweden
104. Professor Malvyn Kalos, Courant Institute for Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012
105. Dr. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
106. Dr. Alan H. Karp, IBM Scientific Center, 1530 Page Mill Road, Palo Alto, CA 94304
107. Dr. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
108. Dr. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550
109. Dr. Kenneth Kennedy, Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77001
110. Dr. Tom Kitchens, ER-7, Applied Mathematical Sciences, Scientific Computing Staff, Office of Energy Research, Office G-437 Germantown, Washington, DC 20545
111. Prof. Clyde P. Kruskal, Department of Computer Science, University of Maryland, College Park, MD 20742
112. Prof. Michael Langston, Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301
113. Dr. Richard Lau, Office of Naval Research, 1030 E. Green Street, Pasadena, CA 91101
114. Dr. Robert L. Launer, Army Research Office, P. O. Box 12211, Research Triangle Park, North Carolina 27709
115. Dr. Scott A. von Laven, Mission Research Corporation, 1720 Randolph Road, SE, Albuquerque, NM 87106-4245
116. Prof. Tom Leighton, Lab for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139
117. Dr. Robert Leland, Oxford University Computing Laboratory, 8-11 Keble Road, Oxford, OX1 3QD, ENGLAND
118. Dr. Randall J. LeVeque, Department of Mathematics, University of Washington, Seattle, WA 98195
119. Dr. John G. Lewis, Boeing Computer Services, P. O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346
120. Dr. Heather M. Liddell, Director, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, ENGLAND
121. Dr. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, Downsview, Ontario, Canada M3J 1P3
122. Dr. Franklin Luk, Electrical Engineering Department, Cornell University, Ithaca, NY 14853
123. Dr. Thomas A. Manteuffel, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
124. Dr. Anita Mayo, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598
125. Dr. James McGraw, Lawrence Livermore National Laboratory, L-306, P. O. Box 808, Livermore, CA 94550
126. Dr. Paul C. Messina, California Institute of Technology, Mail Code 158-79, Pasadena, CA 91125
127. Dr. Cleve B. Moler, MathWorks, 325 Linfield Place, Menlo Park, CA 94025
128. Dr. William A. Mulder, Koninklijke Shell Exploratie en Produktie Laboratorium, Postbus 60, 2280 AB Rijswijk, THE NETHERLANDS
129. Dr. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742
130. Dr. Joseph Oliger, Computer Science Department, Stanford University, Stanford, CA 94305

131. Professor James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

132. Prof. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706

133. Dr. James C. Patterson, Boeing Computer Services, P.O. Box 24346, MS 7L-21, Seattle, WA 98124-0346

134. Dr. Peter C. Patton, Patton Associates, Inc., 101 International Plaza, 7900 International Drive, Minneapolis, MN 55425

135. Dr. Linda R. Petzold, L-316, Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

136. Dr. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650

137. Professor Daniel A. Reed, Computer Science Department, University of Illinois, Urbana, IL 61801

138. Dr. John K. Reid, CSS Division, Building 8.9, AERE Harwell, Didcot, Oxon OX11 0RA, ENGLAND

139. Dr. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907

140. Dr. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore National Laboratory, Livermore, CA 94550

141. Dr. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706

142. Dr. Ahmed H. Sameh, Center for Supercomputing R&D, 1384 W. Springfield Avenue, University of Illinois, Urbana, IL 61801

143. Dr. Jorge Sanz, IBM Almaden Research Center, Department K53/802, 650 Harry Road, San Jose, CA 95120

144. Dr. Robert Schreiber, RIACS, MS 230-5, NASA Ames Research Center, Moffet Field, CA 94035

145. Dr. Martin H. Schultz, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520

146. Prof. Robert B. Schnabel, Department of Computer Science, University of Colorado at Boulder, ECOT 7-7 Engineering Center, Campus Box 430, Boulder, Colorado 80309-0430
147. Dr. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
148. The Secretary, Department of Computer Science and Statistics, The University of Rhode Island, Kingston, RI 02881
149. Prof. Charles L. Seitz, Department of Computer Science, California Institute of Technology, Pasadena, CA 91125
150. Dr. Horst D. Simon, Mail Stop 258-5, NASA Ames Research Center, Moffett Field, CA 94035
151. Dr. William C. Skamarock, 3973 Escuela Court, Boulder, CO 80301
152. Dr. Burton Smith, Teracomp Computer Company, 400 North 34th Street, Suite 300, Seattle, WA 98103
153. Dr. Marc Snir, IBM T.J. Watson Research Center, Department 420/36-241, P. O. Box 218, Yorktown Heights, NY 10598
154. Prof. Larry Snyder, Department of Computer Science, FR-35, University of Washington, Seattle, WA 98195
155. Dr. Danny C. Sorensen, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston, TX 77251
156. Prof. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742
157. Mr. Steven Suhr, Computer Science Department, Stanford University, Stanford, CA 94305
158. Dr. Wei Pai Tang, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2I 3G1
159. Dr. Joseph F. Traub, Department of Computer Science, Columbia University, New York, NY 10027
160. Dr. Lloyd N. Trefethen, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139
161. Prof. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853
162. Dr. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665
163. Mr. Bi R. Vona, Center for Numerical Analysis, RLM 13.150, University of Texas at Austin, Austin, TX 78712
164. Dr. A. J. Wathen, School of Mathematics, University Walk, Bristol BS8 1TW, ENGLAND
165. Dr. Andrew B. White, Los Alamos National Laboratory, P. O. Box 1663, MS-265, Los Alamos, NM 87545


167-176. Office of Scientific & Technical Information, P. O. Box 62, Oak Ridge, TN 37831