Optimal Parallel Algorithms for Solving Tridiagonal Linear Systems

Eunice E. Santos[1]

Department of Electrical Engineering and Computer Science,
Lehigh University, Bethlehem, PA 18015, USA.
Research partially supported by an NSF CAREER Grant.

Abstract. We consider the problem of solving tridiagonal linear systems on parallel distributed-memory machines. We present tight asymptotic bounds for solving these systems on the LogP model using two very common direct methods : odd-even cyclic reduction and prefix summing. For each method, we begin by presenting lower bounds on execution time for solving tridiagonal linear systems. Specifically, we present lower bounds in which it is assumed that the number of data items per processor is bounded, a general lower bound, and lower bounds for specific data layouts commonly used in designing parallel algorithms to solve tridiagonal linear systems. Moreover, algorithms are provided which have running times within a constant factor of the lower bounds provided. Lastly, the bounds for odd-even cyclic reduction and prefix summing are compared.

1 Introduction

Solving tridiagonal linear systems is a basic problem in scientific computing. This problem is especially useful in solving ordinary or partial differential equations which occur naturally in many applications such as fluid dynamics, plasma physics, etc. In this paper, we consider the problem of solving tridiagonal linear systems using direct methods on distributed-memory machines. Although much research has been spent exploring this problem, most deal with designing and analyzing algorithms that solve these systems on specific types of interconnection networks [1, 7, 9, 10] such as hypercube or butterfly. Although some of these algorithms are believed to be efficient, so far no formal proofs are available to substantiate them. One of the first attempts at establishing a lower bound for solving tridiagonal linear systems using odd-even cyclic reduction on distributed-memory machines was advanced by Johnsson [5] and subsequently by Lakshmivarahan and Dhall [8]. However, their results are applicable only to algorithms designed under severe constraints such as a very specific partitioning of tasks onto processors.

The main objective of this paper is to present asymptotically tight bounds on the running time for solving tridiagonal systems which utilize common direct methods; in particular, odd-even cyclic reduction and prefix summing. The results obtained in this paper will provide not only a means for measuring efficiency of existing algorithms but also provide a means of determining what kinds of data layouts and communication patterns are needed to achieve optimal running times.

In order to make our results applicable to a wide spectrum of distributed-memory machines, we will work within LogP [2], a recently proposed model for parallel distributed-memory machines. Important characteristics of parallel machines can be represented using the parameters in the model. An important feature of LogP is that the interconnection network of the machine is modeled by its performance as viewed by the user, rather than its detailed interconnection structure. Algorithms designed on this model are portable from one distributed-memory machine to another and the running times of these algorithms will vary from machine to machine according to the parameter values associated with these machines.

We shall show, among other things, that using common data layouts and straightforward communication patterns do not result in significantly higher complexities than assuming that all processors have access to all data items regardless of communication pattern. In fact, in most cases, common data layouts and straightforward communication patterns can be used to obtain optimal running times. We shall also show that the communication parameters of a network have a significant effect on the complexity of this problem.

The paper is divided as follows. Section 2 contains a description of the LogP model. In Section 3 we consider the odd-even cyclic reduction method for solving tridiagonal systems. We begin by deriving lower bounds on execution time independent of the data layout for this method. Next, we derive lower bounds for data layouts in which the number of data items per processor is bounded. In particular we will show that the skewness of the distribution of data has no significant effect on complexity. We then derive lower bounds for specific data layouts commonly used in designing parallel algorithms for solving tridiagonal linear systems. Lastly, running times for algorithms which are within a constant factor of the lower bounds are provided. Section 4 considers the prefix summing method and derives similar upper and lower bounds on running time. For brevity, proofs and algorithms are not provided in this paper. Many of the proofs and algorithms can be found in [11]. Section 5 gives the conclusion and summary of results.

2 The LogP Model

LogP [2] is a parallel distributed-memory model that specifies the performance characteristics of an interconnection network without describing the structure of the network. Communication between processors is assumed to be point-to-point. The following description uses terminology specific to the problem on hand. The main parameters of the model are:

P: the number of processor/memory modules.

L: an upper bound on the *latency*, or delay, incurred in communicating a message containing a numerical value from its source module to its target module.

o: the *overhead*, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations.

g: the *gap*, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a processor.[1]

The parameters L, o and g are measured as multiples of a processor cycle. A processor cycle is the (unit) time a processor takes to execute an arithmetic operation not requiring communication. Also, at any time step, at most $\lceil L/g \rceil$ messages can be in transit to or from any particular processor. If an attempt to exceed this limit is made by a processor by transmitting a message, the processor stalls until the message can be sent without exceeding the capacity limit. All algorithms discussed in this paper satisfy this capacity constraint, and we do not mention it henceforth.

3 Odd-Even Cyclic Reduction Method

The Problem: Given $M\mathbf{x} = \mathbf{b}$ solve for $\mathbf{x}$, where M is a tridiagonal $N \times N$ matrix, $\mathbf{b} = (b_j)$ is a vector of size N, and $\mathbf{x} = (x_j)$ a vector of size N.

We assume for both the discussion of odd-even cyclic reduction and prefix summing that $1 < P \leq N$. An algorithm is simply a set of arithmetic operations such that each processor is assigned a sequential list of these operations. An initial assignment of data to the processors is called a data layout. A list of message transmissions and receptions between processors is called a communication pattern. These three components (algorithm, data layout, communication pattern) are needed in order to determine running time.

Odd-even cyclic reduction [3, 4, 9] is a recursive method for solving tridiagonal systems of size $N = 2^n - 1$. This method is divided into two parts: reduction and back substitution.

The first step of reduction is to remove each odd-indexed x_i and create a tridiagonal system of size $2^{n-1} - 1$. We then do the same to this new system and continue on in the same manner until we are left with a system of size 1. This requires n phases. We refer to the tridiagonal matrix of phase j as M^j and the vector as $\mathbf{b}^j$. The original M and $\mathbf{b}$ are denoted M^0 and $\mathbf{b}^0$. The three non-zero items of each row i in M^j are denoted l_i^j, m_i^j, r_i^j (left, middle, right). Below are the list of operations needed to determine the items of row i in matrix M^j.

$$e_i^j = -\frac{l_i^{j-1}}{m_{i-2^{j-1}}^{j-1}}, \quad f_i^j = -\frac{r_i^{j-1}}{m_{i+2^{j-1}}^{j-1}}, \quad l_i^j = e_i^j l_{i-2^{j-1}}^{j-1}, \quad r_i^j = f_i^j r_{i+2^{j-1}}^{j-1},$$

$$m_i^j = m_i^{j-1} + e_i^j r_{i-2^{j-1}}^{j-1} + f_i^j l_{i+2^{j-1}}^{j-1}, \quad b_i^j = b_i^{j-1} + e_i^j b_{i-2^{j-1}}^{j-1} + f_i^j b_{i+2^{j-1}}^{j-1}$$

Clearly each system is dependent on the previous systems. In this paper, we assume that if an algorithm employs odd-even cyclic reduction, we assume that a processor computed items of whole rows of a matrix (i.e. the three non-zero data items) and the appropriate item in the vector.

[1] We assume $o \leq g$.

The back substitution phase is initiated after the system of one equation has been determined. We recursively determine the values of the x_i's. The first operation is $x_{2^{n-1}} = \frac{b^{n-1}_{2^{n-1}}}{m^{n-1}_{2^{n-1}}}$. For the remaining variables x_i, let j denote the last phase in the reduction step before x_i is removed then $x_i = \frac{b_i^{j-1} - l_i^{j-1} x_{i-2^{j-1}} - r_i^{j-1} x_{i+2^{j-1}}}{m_i^{j-1}}$.

The serial complexity of this method is $S(N) = 17N - 12\log(N+1) + 11$.

In Section 3.1, we present lower bounds on running time for odd-even cyclic reduction algorithms. Specifically, this section contains the general lower bounds on running time independent of data layout which are applicable to all algorithms utilizing odd-even cyclic reduction, lower bounds on running time for data layouts in which the number of data items initially assigned to a processor is bounded, and lower bounds on running time for common data layouts for this problem. In Section 3.2 we present optimal algorithm running times.

3.1 Lower bounds for Odd-Even Cyclic Reduction

We begin by listing some definitions necessary for the discussion of lower bounds.

Definition 1. The class of all communication patterns is denoted by $\mathcal{C}$. The class of all data layouts is denoted by $\mathcal{D}$. A data layout D is said to be a *single-item layout* if each non-zero matrix-item is initially assigned to a unique processor.

Definition 2. Let A be an odd-even cyclic reduction algorithm. For $i = 1, 2, \cdots n$ and $j = 1, 2, \cdots 2^{n-i+1} - 1$, define $T_{A,D,C}(i,j)$ to be the minimum time at which the items of row j of level i are computed using algorithm A and communication pattern C and assuming data layout D, and define $T_{A,D,C}(i) = min_{1 \leq j \leq 2^{n-i+1}-1} T_{A,D,C}(i,j)$.

It follows that for all odd-even cyclic algorithms A, data layout D, communication pattern C, and any $i < n$, $T_{A,D,C}(i+1) > T_{A,D,C}(i)$.

Definition 3. Let A be an odd-even cyclic reduction algorithm and let $\mathcal{O}$ denote the class consisting of all odd-even cyclic reduction algorithms. For all $i \leq n$, $T_{A,D}(i) = min_{C \in \mathcal{C}} T_{A,D,C}(i)$, $T_A(i) = min_{D \in \mathcal{D}} T_{A,D}(i)$ and $T_{\mathcal{O}}(i) = min_{A \in \mathcal{O}} T_A(i)$.

In the following sections we shall provide lower bounds on $T_{A,D}(n)$ for algorithms $A \in \mathcal{O}$ and certain types of data layouts D. The lower bounds hold regardless of the choice of communication pattern. For simplicity, we state our results in the special case $L = g$ of the LogP model. Although the proofs of the lower bounds are based on the assumption that $o = 0$, they are clearly applicable to arbitrary o.

3.1.1 A General Lower Bound for Odd-Even Cyclic Reduction

In this subsection we assume the data layout is the one in which each processor has a copy of every non-zero entry of M^0 and $\mathbf{b}^0$. We denote this layout by $\bar{D}$. Since $\bar{D}$ is the most favorable data layout, $T_{\mathcal{O}}(i) = min_{A \in \mathcal{O}} T_{A,\bar{D}}(i)$.

Theorem 4. *Let A be an odd-even cyclic reduction algorithm. The following is a lower bound for A regardless of data layout and communication pattern:*

$$\begin{cases} 14(N-n) = \Omega(N) & \textit{if } g \geq 14(N-n) \\ max(g(n-n'), \frac{S(N)}{P}, n) = \Omega(g\log(\frac{N}{g}) + \frac{N}{P} + \log N) & \textit{otherwise} \end{cases}$$

where n' is the smallest integer i such that $g < 14(2^i + i - 1)$.

In Section 3.2 we will provide optimal algorithms, i.e. the running times are within a constant factor of the lower bounds.

3.1.2 Lower Bounds for $\mathcal{O}$ on $\frac{c}{P}$-data layouts

Many algorithms designed for solving tridiagonal linear systems assume that the data layout is single-item and that each processor is assigned roughly $\frac{1}{P}^{th}$ of the rows of M^0 where P is the number of processors available. In this section, we consider single-item data layouts in which each processor is assigned at most a fraction $\frac{c}{P}$ of the rows of M where $1 \leq c \leq \frac{P}{2}$.

Definition 5. Consider c where $1 \leq c \leq \frac{P}{2}$. A data layout D on P processors is said to be a $\frac{c}{P}$-*data layout* if D is single-item, no processor is assigned more than a fraction $\frac{c}{P}$ of the rows of M^0, and each processor is assigned at least one row of M^0. Denote the class of $\frac{c}{P}$ data layouts by $\mathcal{D}(\frac{c}{P})$.

Theorem 6. *If $D \in \mathcal{D}(\frac{c}{P})$, then for any $A \in \mathcal{O}$,*

$$T_{A,D}(n) \geq max(g\lceil\log(P-1)\rceil, g\frac{\lfloor\log\frac{N(P-c)}{P(P-1)}\rfloor - 6}{4}, \frac{S(N)}{P}, n) = \Omega(g\log N + \frac{N}{P}).$$

Analyzing the result given in the above theorem, we see that for any odd-even cyclic algorithm and any $\frac{c}{P}$-data layout, the running time (regardless of communication pattern) is $\Omega(ng + \frac{N}{P})$. Comparing results against the general lower bound, we see that for sufficiently large N, the lower bounds are within constant factors of one another. In Section 3.2, we present algorithms that are within constant factors of the bounds.

Since $c \leq \frac{P}{2}$, every $\frac{c}{P}$-data layout is a $\frac{1}{2}$-data layout. This leads to the following corollary:

Corollary 7. *If $D \in \mathcal{D}(\frac{1}{2})$, then for any $A \in \mathcal{O}$,*

$$T_{A,D}(n) \geq max(g\lceil\log(P-1)\rceil, g\frac{\lfloor\log\frac{N}{P-1}\rfloor - 7}{4}, \frac{S(N)}{P}, n).$$

The complexity of any algorithm $A \in \mathcal{O}$ using a $\frac{c}{P}$-data layout is $\Omega(g\log N + \frac{N}{P})$. We see that the "communication part" of the bound grows only logarithmically in problem size N and is independent of $P(>1)$, whereas the "computation part" grows linear in N and is dependent on P. In addition, although the number of rows assigned to different processors may vary from one to $\frac{1}{2}$ of the total number of initial rows, the above results and the algorithms in Section 3.2 show that the skewness of the distribution of data has no significant effect on complexity.

3.1.3 Lower Bounds for $\mathcal{O}$ on standard data layouts

In this section we present lower bounds on the running time for odd-even cyclic reduction algorithms using specific data layouts commonly used by algorithm designers, namely cyclic and blocked data layouts. Definitions are given below.

Definition 8. A single-item data layout on $(1 \leq)P(\leq N)$ processors $p_1, \cdots p_P$ is *cyclic* if for all $i \leq N$, $r_i^0, m_i^0, l_i^0, b_i^0$ are assigned to processor p_j where $j+1 \equiv i \mod P$. We denote this layout by D_C.

Definition 9. A single-item data layout on $(1 \leq)P(\leq N)$ processors $p_1, \cdots p_P$ is *blocked* if for all $i \leq P$, p_i is assigned the nonzero items in rows $(i-1)\frac{N}{P}+1$ to $i\frac{N}{P}$ of M^0 and $\mathbf{b}^0$. We denote this layout by D_B.

The following definitions are needed for the discussion of lower bounds in this section.

Definition 10. A row j of level i is said to be an *original* row of some processor p if the items of rows $j-2^{i-1}+1$ to $j+2^{i-1}-1$ of level 0 are originally assigned to p. If a row is not an original of p it is said to be a *non-original* row of p.

Definition 11. Two rows j and $j+1$ of level i are referred to as *neighbors*.

Theorem 12. *Let A be an algorithm in $\mathcal{O}$, i.e. A is an odd-even cyclic reduction algorithm. The following are lower bounds assuming the appropriate data layout:*

$$\begin{array}{ll} max(g\lceil\frac{n}{2}\rceil, \frac{S(N)}{P}, n) & \textit{Blocked Data Layout} \\ max(g\lceil\frac{n}{2}\rceil, g\frac{N}{2P}, \frac{S(N)}{P}, n) & \textit{Cyclic Data Layout.} \end{array}$$

Analyzing these lower bounds, we see that blocked and cyclic data layouts have lower bounds $\Omega(g \log N + \frac{N}{P})$ and $\Omega(g(\log N + \frac{N}{P}))$ respectively. In Section 3.2 we provide an algorithm using blocked data layout whose running time is $O(g \log N + \frac{N}{P})$. This clearly shows that the complexity for blocked data layout is as good as and in most cases better than that of cyclic data layout. Therefore, we are able to formally confirm, for the first time, that the widely held belief that cyclic layout is no better than block layout is indeed true for a wide range of parallel machines.

Comparing the complexity of blocked layout with the lower bounds for $\frac{c}{P}$-data layouts, we see that the complexity of blocked layout is within a constant factor of the lower bounds for $\frac{c}{P}$-data layouts. Furthermore, comparing the complexity of blocked layout with the general lower bound (which we show is achievable up to a constant factor in Section 3.2), we see that for sufficiently large n the complexity of blocked layout is within a constant factor of the general lower bound. Therefore we can use the much more realistic blocked data layout rather than $\bar{D}$ and still achieve the lower bounds (up to a constant factor).

3.2 Algorithms and Communication Patterns

We have designed algorithms and communication patterns where when used with the appropriate data layouts have running times matching the lower bounds presented, i.e. the running times differ from the lower bounds by at most a small constant factor. Below is a table of running times. For brevity, algorithms and communication schedules have been omitted. Full algorithm and communication schedules can be found in [11].

Running Time:	**Data Layout**
$10n(g+2o)+\frac{S(N)}{P}=O(g\log N+\frac{N}{P})$	Blocked Layout
$10(n-n')(g+2o)+max(\frac{S(N)}{P},19n)$ $=O(g\log\frac{N}{g}+\frac{N}{P}+\log N)$	$\bar{D}$ – Best Layout
$S(N)=O(N)$	Serial Algorithm

4 Prefix Summing Method

The Prefix Summing Problem: Given N items $s_1, s_2, \cdots, s_N$ and an operator $\bigotimes$, determine $S_1, S_2, \cdots S_N$ where $S_i = x_1 \bigotimes x_2 \bigotimes x_3 \bigotimes \cdots \bigotimes x_i$.

We describe below how we can solve a tridiagonal system of equations by transforming it into a non-commutative prefix summing problem [10]. We assume that $1 < P \leq N$.

We begin by reformulating every equation in the system as a matrix-vector product. Specifically, consider the i^{th} equation $l_i x_{i-1} + m_i x_i + r_i x_{i+1} = b_i$ where $1 \leq i < N$ and $l_1 = 0$. The corresponding matrix-vector product is the following:

$$\begin{pmatrix} x_{i+1} \\ x_i \\ 1 \end{pmatrix} = G_i \begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix} \text{ where } G_i = \begin{pmatrix} -\frac{m_i}{r_i} & -\frac{l_i}{r_i} & \frac{b_i}{r_i} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ (assume } l_1 = 0)$$

From repeated substitution, we see that

$$\begin{pmatrix} x_{i+1} \\ x_i \\ 1 \end{pmatrix} = H_i \begin{pmatrix} x_1 \\ 0 \\ 1 \end{pmatrix} \text{ where } H_i = G_i \cdots G_2 G_1.$$

To solve the tridiagonal system, after H_{N-1} is computed, we solve the following system:

$$\begin{pmatrix} x_N \\ x_{N-1} \\ 1 \end{pmatrix} = H_{N-1} \begin{pmatrix} x_1 \\ 0 \\ 1 \end{pmatrix}, l_N x_{N-1} + m_N x_N = b_N.$$

Therefore, once H_{N-1} is computed we can compute the value of x_1 in 9 steps. We then determine the remaining values of x_i in a similar fashion.

The values of H_i can be computed by prefix summing where $s_i = G_i$ and $\bigotimes$ is 3×3 matrix multiplication (an associative, non-commutative operator).

The serial complexity of this method is $\Theta(N)$.

In Section 4.1, we present a general lower bound (independent of data layout) on running time for prefix summing by considering the complexity of summing. Also, a lower bound for single-item layouts (and therefore a lower bound for $\frac{c}{P}$-data layouts) is provided. Furthermore, a lower bound for blocked data layout is provided. In Section 4.2 we present running times for optimal or near optimal algorithms which use specific types of data layouts for prefix summing. One of the data layouts considered is blocked data layout.

4.1 Lower bounds for Prefix Summing

We present lower bounds on running time for prefix summing algorithms. These lower bounds are based on the lower bounds for summation of N items and for broadcasting an item to P processors. Lower bounds for summation, broadcast and prefix summing are presented in [6, 12]. For simplicity, we state our results in the special case $L = g$ of the LogP model.

Theorem 13. *Any algorithm which solves a tridiagonal linear system of size N using prefix summing requires at least $T_{PS}(N, P)$ where*

$$T_{PS}(N,P) = \begin{cases} N-1, & \text{if } N \le g \\ max(\frac{g}{2}\log(\frac{N}{g}), \frac{N-1}{P}) & \text{if } N \ge g. \end{cases}$$

Theorem 14. *Any algorithm which solves a tridiagonal linear system of size N using prefix summing and using a single-item data layout requires at least*

$$max(g\log(P-1), \frac{N-1}{P}, \frac{g}{2}\log\frac{N}{g}) = \Omega(g\log P(\frac{N}{g})^{\frac{1}{2}} + \frac{N}{P}).$$

Clearly the lower bound for single-item layouts is the lower bound for $\frac{c}{P}$-data layouts

Theorem 15. *Any algorithm which solves a tridiagonal linear system of size N using prefix summing and using a blocked data layout requires at least $T_{PS,D_B}(N, P)$ where*

$$T_{PS,D_B}(N,P) = max(g\log(P-1), \frac{N-1}{P}, \frac{g}{2}\log\frac{N}{g}) = \Omega(g\log P(\frac{N}{g})^{\frac{1}{2}} + \frac{N}{P}).$$

We see that the lower bound for single-item data layouts (and for $\frac{c}{P}$-data layouts) is within a constant factor of the lower bound for blocked layout. Moreover, when N is sufficiently large $T_{PS,D_B}(N,P) = max((g+1)\log(P-1), \frac{N-1}{P}, \frac{g}{2}\log\frac{N}{g}) = \Omega(g\log N + \frac{N}{P})$. Since algorithms are presented in Section 4.2 whose running times are within a constant factor of the lower bounds derived in this section, clearly (1) the complexity for blocked data layout is within a constant factor of the lower bound for $\frac{c}{P}$-data layouts, and (2) for sufficiently large N, the complexity for blocked data layout is within a constant factor of the general lower bound. (3) although the number of rows assigned to different processors may vary from 1 to $N - P + 1$ of the total number of initial rows, the above results and the algorithms presented in the following section show that the skewness of the distribution of data has no significant effect on complexity (in fact, the skewness of distribution that we have proven for prefix summing is greater than the one we've proven for odd-even cyclic reduction).

4.2 Algorithms and Communication Patterns

In this section, we provide a table of running times for algorithms for prefix summing which are within a constant factor of the lower bounds derived in Section 4.1. All of these algorithms are based on the optimal summing, prefix summing and broadcast algorithms presented in [6, 12]. Below is a table of running times. For brevity, algorithms and communication schedules have been omitted. Algorithms and communication schedules can be found in [11].

Running Time:	**Data Layout**
$O(T_{PS,D_B}(N,P))$	Blocked Layout
$O(T_{PS}(N,P))$	$\bar{D}$ - Best Layout
$N-1$	Serial Algorithms

5 Conclusion

We considered the problem of solving tridiagonal linear systems using two very common direct methods: odd-even cyclic reduction and prefix summing. We were able to derive tight asymptotic bounds on the execution time for these problems and provided algorithms which achieve these bounds.

Specifically, we proved that the complexity for solving tridiagonal linear systems for both methods regardless of data layout is $\Theta(g \log \frac{N}{g} + \frac{N}{P})$ for $N = \Omega(g)$ and $\Theta(N)$ for $N = O(g)$. When we added the realistic assumption that the data layouts are single-item and the number of data items assigned to a processor is bounded, we derived the following bounds for odd-even cyclic reduction and prefix summing respectively, $\Theta(g \log N + \frac{N}{P})$ and $\Theta(g \log P(\frac{N}{g})^{\frac{1}{2}} + \frac{N}{P})$. Analyzing these bounds we see that for sufficiently large N the complexity for solving these systems using odd-even cyclic reduction and prefix summing are both $\Theta(g \log N + \frac{N}{P})$. Next comparing these bounds to one another suggests that except for extreme values of P the decision to choose one method over the other may be based on other factors such as numerical stability. We also see that the skewness of data distribution does not significantly affect the complexity of the problem. Specifically, for odd-even cyclic reduction this result is true for single-item data layouts in which no processor is assigned more than $\frac{1}{2}$ of the rows. For prefix summing, the result holds for data layouts which only need to be single-item. In fact, comparing these bounds with the general lower bounds, we see that restricting the proportion of data items assigned to a processor to $\frac{N}{P}$ does not result in a significantly higher complexity than assuming all processors have all the data items for sufficiently large N.

We also derived bounds for blocked and cyclic data layouts. Although it is widely believed that cyclic data layout is no better than blocked data layout, using our results we were able to formally confirm, for the first time, that the "belief" is indeed valid for a wide range of parallel machines.

Lastly, we show that there are algorithms, data layouts, and communication patterns whose running times are within a constant factor of the lower bounds provided. This provides us with the Θ-bounds stated above. To achieve the general lower bound, i.e. the complexity for these methods regardless of data layout, we use $\bar{D}$ the best data layout, i.e. the data layout in which every processor is assigned all the data items. For the $\frac{1}{2}$-data layout lower bound for odd-even cyclic reduction and for the single-item layout lower bound for prefix summing, we use blocked data layout. Clearly blocked data layouts is more realistic than $\bar{D}$ and is easy to assign across processors. Also, since for sufficiently large N these two lower bounds are asymptotic to their respective general lower bounds, this makes the algorithms and communication patterns provided for blocked data layout practical and efficient for these methods.

Since all of the optimal algorithms discussed were variants of standard algorithms using straightforward communication patterns, this shows that it is futile to search for sophisticated techniques, and complicated communication patterns to significantly improve the running times of algorithms which solve tridiagonal systems using odd-even cyclic reduction or prefix summing.

Many of the lower bounds obtained in this paper hold for an extended model with multi-broadcast capability, i.e. the lower bounds hold even under the assumption that any value computed by a processor p is available to that processor immediately and to all other processors $L + 2o$ steps later.

References

1. C. Amodio and N. Mastronardi. A parallel version of the cyclic reduction algorithm on a hypercube. *Parallel Computing*, 19, 1993.
2. D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, E. Santos, K. E. Schauser, R. Subramonian, and T. von Eicken. LogP: A Practical Model of Parallel Computation. *Communications of the ACM*, May 1996.
3. D. Heller. A survey of parallel algorithms in numerical linear algebra. *SIAM J. Numer. Anal.*, 29(4), 1987.
4. A. W. Hockney and C. R. Jesshope. *Parallel Computers*. Adam-Hilger, 1981.
5. S. L. Johnsson. Solving tridiagonal systems on ensemble architectures. *SIAM J. Sci. Stat. Comput.*, 8, 1987.
6. R. M. Karp, A. Sahay, E. E. Santos, and K.E. Schauser Optimal Broadcast and Summation on the LogP Model. In *Proceedings of the Fifth Annual ACM Symposium on Parallel Algorithms and Architectures*, 1993.
7. S. P. Kumar. Solving tridiagonal systems on the butterfly parallel computer. *International J. Supercomputer Applications*, 3, 1989.
8. S. Lakshmivarahan and S. D. Dhall. A Lower Bound on the Communication Complexity for Solving Linear Tridiagonal Systems on Cube Architectures. In *Hypercubes 1987*, 1987.
9. S. Lakshmivarahan and S. D. Dhall. *Analysis and Design of Parallel Algorithms : Arithmetic and Matrix Problems*. McGraw-Hill, 1990.
10. F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*. Morgan Kaufmann, 1992.
11. E. E. Santos. Direct methods for solving tridiagonal linear systems in parallel. Technical Report TR-95-029, International Computer Science Institute, 1995.
12. E. E. Santos. Optimal and efficient parallel algorithms for summing and prefix summing. In *Proceedings of the Eighth Annual IEEE Symposium on Parallel and Distributed Processing*, 1996.