INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Complexity Issues for Solving Triangular Linear Systems in Parallel

Eunice E. Santos*

TR-94-065

December 1994

Abstract

We consider the problem of solving triangular linear systems on parallel distributed-memory machines. Working within the *LogP* model, we present tight asymptotic bounds for solving these systems using forward/backward substitution. Specifically, in this paper we present lower bounds on execution time independent of the data layout, lower bounds for data layouts in which the number of data items per processor is bounded, and lower bounds for specific data layouts commonly used in designing parallel algorithms for this problem. Furthermore, algorithms are provided which have running times within a constant factor of the lower bounds described. Finally, we present a generalization of the lower bounds to banded triangular linear systems.

*Research supported by a DoD-NDSEG Graduate Fellowship and NSF Grant CCR-90-17380.

1 Introduction

In this paper we consider the problem of solving triangular linear systems on distributed-memory machines. While much research has been spent exploring this problem on distributed-memory machines, virtually all of these papers deal with designing and analyzing algorithms on specific types of networks such as the ring or hypercube [5, 6, 12, 13]. Therefore little has been done on developing lower bounds on the running time for solving triangular linear systems for specific networks much less developing lower bounds for distributed-memory machines in general. With such lower bounds, we can determine what types of data distributions are needed to achieve efficient running times.

In order to derive upper and lower bounds on the running time for distributed-memory machines in general, we consider a recently proposed model for parallel computation, called the $LogP$ model[2]. $LogP$ has the important feature that the interconnection network of the machine is modeled by its performance as viewed by the user, rather than its detailed interconnection structure. By using the parameters in $LogP$, many important characteristics of parallel machines can be represented. 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.

In this paper, we focus on deriving lower bounds for the time required to solve triangular linear systems and provide algorithms which achieve these bounds all within the $LogP$ model. The bounds we prove are applicable to algorithms which utilize forward/backward substitution. Using these bounds, we are able to determine which types of data layouts should be assumed in order to achieve an optimal or near-optimal running time.

Among other things, we shall show that the communication parameters of a network have a significant effect on the complexity of this problem. We also show that near-optimal algorithms can be obtained using common data layouts and straightforward communication patterns. Of particular interest, we show that block data layout and block cyclic layouts can incur much higher running times then those of many other common data layouts, such as row/column wrapped. Also, we shall see that restricting the proportion of data items assigned to a processor does not result in a significantly higher complexity than assuming that all processors have access to all data items.

The paper is divided as follows. In Section 2 we describe the $LogP$ model. In Section 3 we list and give a brief discussion of the results obtained in this paper. In Sections 4 and 5 we present tight asymptotic bounds for solving triangular systems using forward/backward substitution. Specifically, we present lower bounds on execution time independent of the data layout, lower bounds for data layouts in which the number of data items per processor is bounded, and lower bounds for specific data layouts commonly used in designing parallel algorithms for this problem, including among others, block data layout and block cyclic layouts of High Performance Fortran [8] and SCALAPACK [4]. Furthermore, algorithms are provided which have running times within a constant factor of the lower bounds described. Finally, we present a generalization of the lower bounds to banded triangular linear systems. Section 6 gives the conclusion and summary of results.

2 The $LogP$ Model

$LogP$ is a model of a distributed-memory multiprocessor in which processors communicate by point-to-point messages [2]. The model specifies the performance characteristics of the interconnection network, but does not describe the structure of the network. We have tailored the description

of the model using terminology specific to the problem of solving triangular systems. 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. [Note : $o \leq g$]

Arithmetic operations not requiring communication between processors execute in unit time (a processor cycle). The parameters L, o and g are measured as multiples of the processor cycle. Furthermore, it is assumed that the network has a *finite capacity*, such that at most $\lceil L/g \rceil$ messages can be in transit from any processor or to any processor at any time. If a processor attempts to transmit a message that would exceed this limit, it stalls until the message can be sent without exceeding the capacity limit. All algorithms discussed in this paper satisfy the capacity constraint of the $LogP$ model, and we do not mention it henceforth.

3 Discussion of Results

In this section we discuss the results obtained in the following sections of this paper. As stated before, we are deriving upper and lower bounds on the running time of substitution algorithms.

Three components are needed in order to determine running time : the algorithm, the data layout, and the communication pattern. We view an algorithm simply as a set of arithmetic operations where each processor is assigned a sequential list of these operations. A data layout is the initial assignment of data to the processors. A communication pattern is a list of message transmissions and receptions between processors.

Clearly any substitution algorithm must have at least n^2 operations and therefore requires a minimum of $\frac{n^2}{P}$ time steps.

We begin by considering data layouts in which (1) no data item is assigned to more than one processor and (2) no processor is assigned more than half of the data items. Clearly the most common data layouts used by algorithm designers fall into this category. We show that any substitution algorithm using such a data layout will have a running time of at least $max([(\frac{\sqrt{2}-1}{\sqrt{2}})n + 1]g, \frac{n^2}{P})$. In Subsection 4.5 we present substitution algorithms using these types of data layouts and straightforward communication patterns. The running times are within constant factors of the lower bound given above.

We next derive bounds on the running time independent of the type of data layout used. We show that when $n \leq \lfloor \sqrt{g} \rfloor$ the lower bound is n^2, when $\lfloor \sqrt{g} \rfloor \leq n \leq g$ the lower bound is $\lceil \frac{n^2+4n-3}{4} \rceil$ and when $n \geq g$ the lower bound is $max(\lceil \frac{n}{4} \rceil g, \frac{n^2}{P})$. In Subsection 4.5 we provide algorithms whose running times are within a constant factor of these bounds. In fact, since the straightforward uniprocessor algorithm achieves a running time of n^2, the case $n \leq \lfloor \sqrt{g} \rfloor$ achieves a tight bound on running time. Analyzing these bounds, we see that when $n \leq g$ there is no significant benefit in using multiple processors. When $n \geq g$, we see that near-optimal running times can be achieved using straightforward communication patterns.

The last set of lower bounds for triangular systems are for very commonly-used data layouts. In particular we consider row/column wrapped, row/column contiguous, block and block cyclic data layouts. We formally define these data layouts in Subsection 4.3. We show that for the row/column wrapped and row/column contiguous, the running times are $\Theta(ng + \frac{n^2}{P})$ and the communication patterns are simple. The precise values of these bounds are given in Subsections 4.3- 4.5. We show that block and block-cyclic layouts have higher complexities.

Lastly we consider the problem of solving banded triangular systems. Again, we assume the algorithms all utilize substitution. As before, we are able to deduce that there is no significant benefit in not choosing a straightforward communication pattern.

4 Forward/Backward Substitution for Solving Triangular Linear Systems

We solve triangular linear systems using the forward/backward substitution method. Our decision to focus on this method is based on the fact that while there have been many methods for solving triangular linear systems [1, 7, 10, 14], many of these methods have been shown to be numerically unstable and/or require a number of arithmetic operations which is not optimal [3, 9]. On the other hand, substitution algorithms have been shown to have perfect numerical stability and clearly use the minimum number of arithmetic operations [9]. In addition, substitution is a standard method utilized by algorithm designers.

The Problem : Given $T\mathbf{x} = \mathbf{b}$ solve for $\mathbf{x}$, where $T = (a_{i,j})$ is a (lower) triangular $n \times n$ matrix, $\mathbf{b} = (b_j)$ is a vector of size n, and $\mathbf{x} = (x_j)$ a vector of size n.

Definition 4.1 *For all $i \leq n$, a* psum *of x_i has the following form :*

$$\frac{1}{\bar{a}_{i,i}}[\bar{b}_i - \sum_{j=1}^{i-1} a_{i,j}\bar{x}_j] \quad or \quad \frac{1}{\bar{a}_{i,i}} \sum_{j=1}^{i-1} a_{i,j}\bar{x}_j$$

where $\bar{a}_{i,i} = a_{i,i}$ or 1, $\bar{b}_j = b_j$ or 0, and $\bar{x}_j = x_j$ or 0.

We now define our class of algorithms. An algorithm is viewed as a set of arithmetic operations where each processor is assigned a sequential list of these operations. Since we focus on the forward/backward substitution method, we clearly deal only with algorithms whose arithmetic operations create linear combinations (psums) of x_j's either by (1) multiplying/dividing data items or (2) using an arithmetic operation on two psums to form a new psum. Below is a more formal definition.

Definition 4.2 *We say an algorithm A is a* substitution algorithm *if each arithmetic operations is one of the following : (1) multiplying some $a_{i,j}$ and x_j where $j < i$, (2) adding/subtracting two psums of some x_i resulting in another psum of x_i, or (3) dividing a psum of some x_i by $a_{i,i}$ to form another psum of x_i. We denote the class of substitution algorithms by $\mathcal{A}$.*

It follows from the above definition that processors can only receive or transmit the following values : some b_j or $a_{i,j}$ or a psum of some x_j. Moreover, since we are working in the $LogP$ model, we assume that

1. if processor p transmit a message to $\bar{p}$ at time t then

(a) it takes o steps for p to place the message into the network,

(b) at time $t+o$ to $t+o+L$, the message is in transit in the network, and

(c) it takes o steps for $\bar{p}$ to retrieve the message.

2. if processor p receives (transmits) a message at time t, p cannot receive (transmit) another message until time $t+g$ and cannot perform any type of operation until time $t+o$.

All the algorithms we present satisfy all the constraints of the $LogP$ model. Moreover, our lower bounds hold even under the stronger assumption that any value computed by a processor p is available to that processor immediately and to all other processors L steps later.

Definition 4.3 *A* data layout *is the initial distribution of data onto the processors. The class of all data layouts is denoted $\mathcal{D}$. A data layout D is said to be a* single-item layout *if each matrix entry $a_{i,j}$ is initially assigned to a unique processor.*

Definition 4.4 *For any algorithm $A \in \mathcal{A}$ and data layout D, define $T_{A,D}(x_i)$, $i = 1, 2, \cdots n$ to be the time at which x_i is computed using algorithm A and assuming data layout D. (If x_i is computed more than once, $T_{A,D}(x_i)$ is the time at which x_i was first computed.)*

It follows that for all $A \in \mathcal{A}$ and $D \in \mathcal{D}$, and any $i < n$, $T_{A,D}(x_{i+1}) > T_{A,D}(x_i)$.

Definition 4.5 *Let $A \in \mathcal{A}$. For all $i < n$, $T_A(x_i) = min_{D \in \mathcal{D}} T_{A,D}(x_i)$ and $T_{\mathcal{A}}(x_i) = min_{A \in \mathcal{A}} T_A(x_i)$ (i.e. $T_{\mathcal{A}}(x_i)$ is the minimum time needed to compute x_i by any algorithm in the class $\mathcal{A}$ regardless of data layout).*

In the following sections we shall prove lower bounds on $T_{A,D}(x_n)$ for algorithms $A \in \mathcal{A}$ 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. In Section 4.1 we prove lower bounds on $\mathcal{A}$ assuming D is a single-item data layout in which the number of data items a processor is initially assigned is bounded. In Section 4.2 we assume $\bar{D}$ is the data layout in which every processor has a copy of every data item (i.e. all $a_{i,j}$ and b_j). We present a lower bound on $T_{A,\bar{D}}(x_n)$. Since $\bar{D}$ is the most favorable layout, there is no algorithm $A \in \mathcal{A}$ and data layout $D \in \mathcal{D}$ which can complete earlier than this bound. In Section 4.3 we prove some lower bounds for $A \in \mathcal{A}$ where D is a standard data layout such as the row/column wrapped data layout. Although the proofs of the lower bounds In Sections 4.1- 4.3 are based on the assumption that $o = 0$, they are clearly applicable to arbitrary o. In Section 4.4 we present an extension of the class of algorithms where the lower bounds are still applicable. In Section 4.5, we discuss algorithms for various data layouts. These algorithms all run within a constant factor of the lower bounds shown in this section.

4.1 Lower bounds for $A \in \mathcal{A}$ where D is a $\frac{c}{P}$ - data layout

Many algorithms designed for solving triangular systems assume that the data layout is single-item and that each processor is assigned roughly $\frac{1}{P}^{th}$ of the data items of T where P is the number of processors available [5, 6, 12, 13]. In this subsection we consider single-item data layouts where each processor can be assigned at most a fraction $\frac{c}{P}$ of the data items of T where $1 \leq c \leq \frac{P}{2}$.

Definition 4.6 *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 and no processor is assigned more than a fraction $\frac{c}{P}$ of the $a_{i,j}$'s of T. Denote the class of $\frac{c}{P}$ data layouts by $\mathcal{D}(\frac{c}{P})$.*

In this section we prove the following result :

Theorem 4.1 *If $D \in \mathcal{D}(\frac{c}{p})$, then for any $A \in \mathcal{A}$,*

$$T_{A,D}(x_n) \geq max([(1 - \sqrt{\frac{c}{P}})n + 1]g, \frac{n^2}{P}).$$

In order to present the proof of Theorem 4.1, the following definitions are needed.

Definition 4.7 *Let D be a single-item layout, then for all $i \leq n$, define $p_{i,D}$ to be the processor which is initially assigned item $a_{i,i}$.*

Definition 4.8 *For any algorithm $A \in \mathcal{A}$ and data layout D, define $\forall i \leq n$, $p_{A,D}(i)$ to be the processor(s) which computes x_i in time $T_{A,D}(x_i)$.*

Definition 4.9 *Let $i_{n,\frac{c}{P}}$ be the smallest integer i such that $\frac{1}{2}i(i+1) > \frac{n(n+1)}{2}\frac{c}{P}$.*

Clearly $i_{n,\frac{c}{P}} - 1$ is the largest possible number such that all data items in rows 1 to $i_{n,\frac{c}{P}} - 1$ of T are assigned to a single processor.

It follows immediately that $\sqrt{\frac{c}{P}}n - 1 < i_{n,\frac{c}{P}} \leq \sqrt{\frac{c}{P}}n$.

Proof: (Theorem 4.1) We begin by showing that for all $A \in \mathcal{A}$ and $D \in \mathcal{D}(\frac{c}{P})$, if $\exists i \geq i_{n,\frac{c}{P}}$ such that

- $T_{A,D}(x_i) < (i - i_{n,\frac{c}{P}} + 1)g$, and
- for all j where $i_{n,\frac{c}{P}} \leq j < i$, $T_{A,D}(x_j) \geq (j - i_{n,\frac{c}{P}} + 1)g$

then $x_1, \cdots, x_i$ must all be received or computed by $p_{A,D}(i)$.

We prove this using contradiction. Suppose k is the highest index in which x_k is not received/computed in $p_{A,D}(i)$. This implies that $p_{A,D}(i)$ must receive at least $i - k$ messages after time $T_{A,D}(x_k)$. This requires at least $(i - i_{n,\frac{c}{P}} + 1)g$ time steps which is a contradiction.

Next we prove that for all $A \in \mathcal{A}$, $D \in \mathcal{D}(\frac{c}{p})$ and $i \geq i_{n,\frac{c}{P}}$, $T_{A,D}(x_i) \geq (i - i_{n,\frac{c}{P}} + 1)g$.

The proof is by induction. The base case is evident. For the induction step, suppose $T_{A,D}(x_{i+1})$ $< (i - i_{n,\frac{c}{P}} + 2)g$ where $i+1 \geq i_{n,\frac{c}{P}}$ and for all j where $i_{n,\frac{c}{P}} \leq j < i+1$, $T_{A,D}(x_j) \geq (j - i_{n,\frac{c}{P}} + 1)g$. From above, $x_1, \cdots, x_{i+1}$ must be computed/received by $p_{A,D}(i+1)$. Since $p_{A,D}(i+1)$ can receive at most $i - i_{n,\frac{c}{P}} + 1$ messages, the numbers of items initially assigned to $p_{A,D}(i+1)$ must be at least $\frac{n(n+1)}{2}\frac{c}{P}$ which is a contradiction. Lastly, it is clear that $\frac{n^2}{P}$ is a lower bound. □

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 4.1 *If $D \in \mathcal{D}(\frac{c}{p})$, then for any $A \in \mathcal{A}$,*

$$T_{A,D}(x_n) \geq max([(\frac{\sqrt{2}-1}{\sqrt{2}})n + 1]g, \frac{n^2}{P}).$$

The complexity of any algorithm A in our class using a $\frac{c}{P}$-data layout is $\Omega(ng + \frac{n^2}{P})$; we see that the "communication part" of the bound, ng, is independent of $P(> 1)$. In addition, although the number of data items assigned to different processors may vary from no data items to $\frac{1}{2}$ of the total number of data items, the above results and the algorithms provided in Section 4.5 show that the skewness of the distribution of data has no significant effect on the complexity.

4.2 A general lower bound for all $A \in \mathcal{A}$

In the previous subsection we proved lower bounds for data layouts in which the number of data items assigned per processor is bounded. In this subsection we assume the data layout is the one in which each processor has a copy of every $a_{i,j}$ and b_j where $i, j \leq n$. We denote this layout by $\bar{D}$. Since $\bar{D}$ is the most favorable data layout, $T_{\mathcal{A}}(x_i) = min_{A \in \mathcal{A}} T_{A,\bar{D}}(x_i)$.

In this section, we prove the following result :

Theorem 4.2

$$T_{\mathcal{A}}(x_n) \geq \begin{cases} n^2 & if\ n \leq \lfloor \sqrt{g} \rfloor \\ \lceil \frac{n^2+4n-3}{4} \rceil & if\ \lfloor \sqrt{g} \rfloor \leq n \leq g \\ max(\lceil \frac{n}{4} \rceil g, \frac{n^2}{P}) & if\ n \geq g \end{cases}$$

Proof: We begin by proving that for all $A \in \mathcal{A}$ and for all i such that $i \leq min(n, \lfloor \sqrt{g} \rfloor)$, $T_{A,\bar{D}}(x_i) \geq i^2$.

The proof is by contradiction. Assume $\exists i \leq min(n, \lfloor \sqrt{g} \rfloor)$ such that $T_{A,\bar{D}}(x_i) < i^2$. Since $i^2 \leq g$ thus each $x_1, \cdots x_i$ must be computed directly and locally in the processor. This takes at least i^2 arithmetic steps which is a contradiction.

We now observe that for all $A \in \mathcal{A}$, if A has the following properties :

1. $\exists j$ such that

$$T_{A,\bar{D}}(x_j) < \begin{cases} \lceil \frac{j^2+4j-3}{4} \rceil & \text{if } j \leq g \\ \lceil \frac{j}{4} \rceil g & \text{if } j \geq g \end{cases}$$

2. $\forall i < j$,

$$T_{A,\bar{D}}(x_i) \geq \begin{cases} \lceil \frac{i^2+4i-3}{4} \rceil & \text{if } i \leq g \\ \lceil \frac{i}{4} \rceil g & \text{if } i \geq g \end{cases}$$

then $x_1, x_2, \cdots x_j$ must all be computed or received by $p_{A,\bar{D}}(j)$.

We prove this observation using contradiction. Let i be the highest index where x_i is not received/computed in $p_{A,\bar{D}}(j)$. This means that $p_{A,\bar{D}}(j)$ must receive at least $j - i$ message after time $T_{A,\bar{D}}(x_i)$. But $T_{A,\bar{D}}(x_i) + (j-i)g \geq T_{A,\bar{D}}(x_j)$ which is a contradiction.

Using the above observation, we next show that for all $A \in \mathcal{A}$ and for all i such that $\lfloor \sqrt{g} \rfloor \leq i \leq min(n, g)$, $T_{A,\bar{D}}(x_i) \geq \lceil \frac{i^2+4i-3}{4} \rceil$.

The proof uses induction. The base case is evident. For the induction step, suppose $T_{A,\bar{D}}(x_{i+1}) < \lceil \frac{(i+1)^2+4(i+1)-3}{4} \rceil$. From the above observation, $x_1, x_2, \cdots x_{i+1}$ must all be either received or computed by $p_{A,\bar{D}}(i+1)$. Since $\lceil \frac{(i+1)^2+4(i+1)-3}{4} \rceil \leq \frac{i+5}{4} g$, thus $p_{A,\bar{D}}(i+1)$ receives less than $\frac{i+5}{4}$ messages. Therefore $p_{A,\bar{D}}(i+1)$ must scan at least $\lceil \frac{(i+1)^2+4(i+1)-3}{4} \rceil$ items which is a contradiction.

Using a proof similar to the one above, we see that for all $A \in \mathcal{A}$ and for all i such that $g \leq i \leq n$, $T_{A,\bar{D}}(x_i) \geq \lceil \frac{i}{4} \rceil g$.

From the results obtained so far and from the fact that $\frac{n^2}{P}$ is a lower bound, we see that for any algorithm A in the class $\mathcal{A}$,

$$T_{A,\bar{D}}(x_n) \geq \begin{cases} n^2 & \text{if } n \leq \lfloor\sqrt{g}\rfloor \\ \lceil\frac{n^2+4n-3}{4}\rceil & \text{if } \lfloor\sqrt{g}\rfloor \leq n \leq g \\ max(\lceil\frac{n}{4}\rceil g, \frac{n^2}{P}) & \text{if } n \geq g \end{cases}$$

Since $\bar{\mathcal{D}}$ is the best data layout possible, no other data layout could have a lower bound less than the lower bound for $\bar{\mathcal{D}}$. Therefore, these bounds also hold for $T_{\mathcal{A}}(x_n)$. □

Comparing the bounds from Sections 4.1- 4.2 and considering the algorithms in Section 4.5, we see that for $n \geq g$, the restriction to $\frac{c}{P}$-data layouts increases complexity by only a constant factor.

4.3 Lower Bounds for $\mathcal{A}$ on standard data layouts

In this section we present lower bounds on the running time for algorithms using specific data layouts which are commonly used by algorithm designers. The data layouts we consider are the following : row/column wrapped, row/column contiguous, block decomposition, and block-cyclic. Formal definitions of these data layouts are given below.

Definition 4.10 *The* row[column] wrapped data layout *on $P(\leq n)$ processors $\rho_1, \rho_2 \cdots \rho_P$ is defined as follows : for all $i \leq n$, $a_{i,1}, \cdots a_{i,i}$ $[a_{1,i}, \cdots a_{i,i}]$ is assigned to processor p_j where $j+1 \equiv i \bmod P$. We denote this layout by $D_{R_w}[D_{C_w}]$.*

Definition 4.11 *A single-item data layout on $P = \frac{\bar{P}(\bar{P}+1)}{2}$ (where $\bar{P} \leq n$) processors is said to be a* block *data layout if each processor is given a contiguous block of T consisting of a square matrix of size $\frac{n}{\bar{P}}$. We denote this data layout by D_{BD}.*

Definition 4.12 *A single-item data layout on $P(\leq n^2)$ processors is said to be a* (square) block-cyclic *data layout if there exist a $1 \leq K \leq \frac{n}{\sqrt{P}}$: T is divided into contiguous square blocks of size $\frac{n}{K}$. Furthermore, each such block is divided into smaller contiguous square sub-blocks of size $\frac{n}{K\sqrt{P}}$. Each processor is then assigned a sub-block of each block of size $\frac{n}{K\sqrt{P}}$ such that each sub-block is in the same position for each block. We denote this data layout by $D_{K,BC}$.*

Definition 4.13 *A single-item data layout on $P(\leq n)$ processors $p_1, \cdots p_P$ is* row[column] contiguous *if for all $i \leq P$, p_i is assigned the matrix items in rows [columns] $(i-1)\frac{n}{P}+1$ to $i\frac{n}{P}$. We denote this layout by $D_{R_c}[D_{C_c}]$.*

We show that block decomposition and clock-cyclic data layouts have higher lower bounds than row/column wrapped or row/column contiguous.

In this section we prove the following result :

Theorem 4.3 *Let A be an algorithm in the class $\mathcal{A}$. The following are true :*

$$T_{A,D_{R_w}}(x_n) \geq max([\lceil\tfrac{P-1}{P}n\rceil - 1]g, \tfrac{n^2}{P}) \qquad \textit{Row Wrapped Layout}$$

$$T_{A,D_{C_w}}(x_n) \geq max([\lceil\tfrac{P-1}{P}n\rceil - 1]g, \tfrac{n^2}{P}) \qquad \textit{Column Wrapped Layout}$$

$$T_{A,D_{BD}}(x_n) \geq max((n - \tfrac{n}{\bar{P}} - 1)g, \tfrac{n^2}{P}, min(\tfrac{n(\bar{P}-1)}{4\bar{P}}(\tfrac{n}{\bar{P}}+1), \tfrac{n(\bar{P}-1)}{4\bar{P}^2}(\tfrac{n}{\bar{P}}+1)g)) \qquad \textit{Block Data Layout}$$

$$T_{A,D_{K,BC}}(x_n) \geq max((n - \tfrac{n}{\sqrt{P}} - 1)g, \tfrac{n^2}{P}, min(\tfrac{n(K\sqrt{P}-1)}{K\sqrt{P}}(\tfrac{n}{K\sqrt{P}}+1), \tfrac{n(K\sqrt{P}-1)}{4KP}(\tfrac{n}{K\sqrt{P}}+1)g)) \qquad \textit{Block-Cyclic Layout}$$

$$T_{A,D_{R_c}}(x_n) \geq max(\lceil\tfrac{P-1}{2P}n\rceil g, \tfrac{n^2}{P}) \qquad \textit{Contiguous Row Layout}$$

$$T_{A,D_{C_c}}(x_n) \geq max(\lceil\tfrac{P-1}{2P}n\rceil g, \tfrac{n^2}{P}) \qquad \textit{Contiguous Column Layout}$$

Proof: (1) We begin by showing that for all $A \in \mathcal{A}$ and D is one of either row or column wrapped. if $\exists i$ such that

- $T_{A,D}(x_i) < (\lceil \frac{P-1}{P} i \rceil - 1)g$, and
- for all j where $j < i$, $T_{A,D}(x_j) \geq (\lceil \frac{P-1}{P} j \rceil - 1)g$

then $x_1, \cdots, x_i$ must all be received or computed by $p_{A,D}(i)$.

We prove this using contradiction. Suppose k is the highest index in which x_k is not received/computed in $p_{A,D}(i)$. This implies that $p_{A,D}(i)$ must receive at least $i-k$ messages after time $T_{A,D}(x_k)$. This requires at least $(\lceil \frac{P-1}{P} i \rceil - 1)g$ time steps which is a contradiction.

Next we prove that for all $A \in \mathcal{A}$ and D is one of either row or column wrapped then $T_{A,D}(x_i) \geq (\lceil \frac{P-1}{P} i \rceil - 1)g$.

The proof is by induction. The base case is evident. For the induction step, suppose $T_{A,D}(x_{i+1}) < (\lceil \frac{P-1}{P}(i+1) \rceil - 1)g$ and for all j where $j < i+1$, $T_{A,D}(x_j) \geq (\lceil \frac{P-1}{P} j \rceil - 1)g$. From above, $x_1, \cdots, x_{i+1}$ must be computed/received by $p_{A,D}(i+1)$. WLOG assume $p_{A,D}(i+1) = p_{m,D}$. Consider x_s where $p_{s,D} \neq p_{m,D}$. If x_s is computed by $p_{m,D}$, then $p_{m,D}$ must receive $a_{s,s}$. Else x_s is received by $p_{m,D}$. Since there are $\lceil \frac{P-1}{P}(i+1) \rceil - 1$ such x_s's, $p_{m,D}$ must receive at least $\lceil \frac{P-1}{P}(i+1) \rceil - 1$ messages which is a contradiction. Lastly, it is clear that $\frac{n^2}{P}$ is a lower bound.

(2) Using a similar argument as in (1), we can prove that $\max(\frac{n^2}{P}, (n - \frac{n}{P} - 1)g)$ is a lower bound for both block data layouts and block-cyclic layouts.

(3) We now prove that $\min(\frac{n(\bar{P}-1)}{4\bar{P}^2}(\frac{n}{\bar{P}}+1)g, \frac{n(\bar{P}-1)}{4\bar{P}}(n\bar{P}+1))$ is a lower bound for block data layout.

Suppose $\frac{n(\bar{P}-1)}{4\bar{P}^2}(\frac{n}{\bar{P}}+1)g < \frac{n(\bar{P}-1)}{4\bar{P}}(n\bar{P}+1)$. Thus $g < \bar{P}$. We prove using induction that $T_{A,D_{BD}}(x_{\frac{ni}{\bar{P}}+1}) \geq \frac{ni}{4\bar{P}^2}(\frac{n}{\bar{P}}+1)g$ for $i < \bar{P}$.

The base case is evident. For the induction case, consider $i = m$. Processor $p_{\frac{nm}{\bar{P}}, D_{BD}}$ has $\frac{n}{2\bar{P}}(\frac{n}{\bar{P}}+1)$ items initially assigned to it. For each item assigned to this processor, it can either send the item to be computed in another processor or keep it to be used in computing the x_t's. If it sends out at least $\frac{nm}{4\bar{P}^2}(\frac{n}{\bar{P}}+1)$ messages then we achieve our bound. Suppose the processor sends out s messages where $s < \frac{nm}{4\bar{P}^2}(\frac{n}{\bar{P}}+1)$. Since any item which is not sent must now be computed within this processor and these items cannot be computed until after time $T_{A,D_{BD}}(x_{\frac{n(m-1)}{\bar{P}}+1})$. Therefore, we see that $T_{A,D_{BD}}(x_{\frac{nm}{\bar{P}}+1}) \geq \frac{nm}{4\bar{P}^2}(\frac{n}{\bar{P}}+1)g$.

The proof that $\frac{n(\bar{P}-1)}{4\bar{P}}(n\bar{P}+1)$ is a lower bound when $\frac{n(\bar{P}-1)}{4\bar{P}^2}(\frac{n}{\bar{P}}+1)g \geq \frac{n(\bar{P}-1)}{4\bar{P}}(n\bar{P}+1)$ follows using a similar argument.

(4) The lower bound for block cyclic layouts can be proved using similar arguments as those given for block layouts.

(5) Suppose D is row contiguous and the lower bound stated above does not hold. Consider $p_{n,D}$. Clearly, less than $\lceil \frac{P-1}{2P} n \rceil$ messages were sent out by $p_{n,D}$. But this implies that at least $\lceil \frac{P-1}{2P} n \rceil$ x_i's where $i \leq \frac{P-1}{P} n$ must be received or computed by $p_{n,D}$. Since $\forall i \leq \frac{P-1}{P} n$, $p_{i,D} \neq p_{n,D}$, $p_{n,D}$ must receive $\lceil \frac{P-1}{2P} n \rceil$ messages which is a contradiction.

The proof also applies to column contiguous except the processor we consider is $p_{1,D}$. □

Analyzing these lower bounds, we see that row/column wrapped and row/column contiguous have lower bounds of $\Omega(max(ng, \frac{n^2}{P}))$. We have shown that the lower bound for block data layout is $\Omega(max(ng, \frac{n^2}{P}, min(\frac{n^2}{P}g, \frac{n^2}{\sqrt{P}})))$. Clearly $max(ng, \frac{n^2}{P}) < max(ng, \frac{n^2}{P}, min(\frac{n^2}{P}g, \frac{n^2}{\sqrt{P}}))$ only when

$P < min(n, \frac{n^2}{g^2})$. Now, the lower bound for block cyclic layout is $\Omega(max(ng, \frac{n^2}{P}, min(\frac{n^2}{KP}g, \frac{n^2}{K\sqrt{P}})))$. Again, clearly $max(ng, \frac{n^2}{P}) < max(ng, \frac{n^2}{P}, min(\frac{n^2}{KP}g, \frac{n^2}{K\sqrt{P}}))$ when $K < min(\sqrt{P}, \frac{n}{P}, \frac{n}{g\sqrt{P}}, g)$.

4.4 Extended Class of Algorithms $\mathcal{B}$

In this subsection we define an extended class of algorithms $\mathcal{B}$ which we call *no-cost inference algorithms* to which all the results obtained in the previous subsections are applicable. The class $\mathcal{B}$ contains all algorithms in which a processor is allowed to infer the value of a variable x_i from the psums it has computed or received and from data items in its local memory. Such an inference is possible if and only if x_i can be expressed as a rational combination of such psums and data items. When proving lower bounds on complexity we do not charge any cost for such an inference (see Example 4.1).

Example 4.1 *Suppose s_1 and s_2 are psums received or computed by processor p where*

$$s_1 = b_{i_1} - a_{i_1,j_1}x_{j_1} - a_{i_1,j_2}x_{j_2}, \quad s_2 = a_{i_2,j_1}x_{j_1} + a_{i_2,j_2}x_{j_2}.$$

If $b_{i_1}, a_{i_1,j_1}, a_{i_1,j_2}, a_{i_2,j_1}, a_{i_2,j_2}$ are in the local memory of p then we assume that p can infer (at no cost) the values of x_{j_1} and x_{j_2}.

4.5 Algorithms

In this section we provide algorithms and communication patterns where when used with the appropriate data layout has running times nearly matching the lower bounds presented in Sections 4.1- 4.3, i.e. the running times differ from the lower bounds by at most a constant factor. We assume that the items of vector **b** are available to all the processors. In Section 4.6 we discuss why this assumption does not affect the complexity of the problem beyond a constant factor. We assume that the value of x_i will be in the variable $b[i]$ which initially contained the value b_i.

Algorithm $A_{D_{RW}}$ and Communication Pattern $C_{D_{RW}}$ for Row-Wrapped Data Layout

Every processor $p_{i,D_{RW}}$ (where $1 \leq i \leq P$) do in parallel :
  for $l = 1$ to $n - P + i$ do
    if $p_{l,D_{RW}} \neq p_{i,D_{RW}}$ then
      receive $b[l]$ from $p_{i-1,D_{RW}}$ /*if $i = 1$ then $p_{i-1,D_{RW}} = p_{P,D_{RW}}$*/
      send $b[l]$ to $p_{i+1,D_{RW}}$ /*if $i = P$ then $p_{i+1,D_{RW}} = p_{1,D_{RW}}$*/
    else
      $b[l] \leftarrow \frac{b[l]}{a_{l,l}}$
      send $b[l]$ to $p_{i+1,D_{RW}}$
    for $s = 0$ to $\frac{n}{p} - 1$ do
      if $a_{Ps+i} \neq 0$ then
        $b[Ps + i] \leftarrow b[Ps + i] - a_{Ps+i,l}b[l]$

Running Time : $P\sum_{j=1}^{\frac{n}{P}} max(g, 2j + o + 2) + (n - 1)(2o + g + 1) = O(ng + \frac{n^2}{P})$

Algorithm $A_{D_{CW}}$ and Communication Pattern $C_{D_{CW}}$ for Column-Wrapped Data Layout

Every processor $p_{i,D_{CW}}$ $(1 \leq i \leq P)$ do in parallel :
  for $l = 0$ to $\frac{n}{P} - 1$ do
    if $l \neq 0$ then

for $m = i+1$ to P **do**
receive $temp[Pl+i]$ from $p_{m,D_{CW}}$
$b[Pl+i] \leftarrow b[Pl+i] - temp[Pl+i]$
for $m = 1$ to $i-1$ **do**
receive $temp[Pl+i]$ from $p_{m,D_{CW}}$
$b[Pl+i] \leftarrow b[Pl+i] - temp[Pl+i]$
$b[Pl+i] \leftarrow \frac{b[Pl+i]}{a_{Pl+i,Pl+i}}$
for $m = Pl+i+1$ to n **do**
if $l = 0$ then
$temp[m] \leftarrow a_{Pl+i,m} b[Pl+i]$
else
$temp[m] \leftarrow temp[m] + a_{Pl+i,m} b[Pl+i]$
if $m < P(l+1)+i$ then
send $temp[m]$ to $p_{m,D_{CW}}$

Running Time : $\frac{2n^2}{P} + (3n - \frac{n}{P})(g+2o) + \sum_{i=1}^{\frac{n}{P}} \sum_{j=1}^{P} max(iP-j, g) = O(ng + \frac{n^2}{P})$

Algorithm $A_{D_{RC}}$ and Communication Pattern $C_{D_{RC}}$ for Contiguous Row Data Layout

Every processor $p_{\frac{ni}{P}+1,D_{RC}}$ do in parallel :
for $l = 1$ to $\frac{ni}{P}$ do
receive $b[l]$ from $p_{\frac{n(i-1)}{P}+1,D_{RC}}$
for $m = \frac{ni}{P}+1$ to $\frac{n(i+1)}{P}$ do
$b[m] \leftarrow b[m] - a_{m,l} b[l]$
send $b[l]$ to $p_{\frac{n(i+1)}{P}+1,D_{RC}}$
for $l = \frac{ni}{P}+1$ to $\frac{n(i+1)}{P}$ do
for $m = \frac{ni}{P}+1$ to $l-1$ do
$b[m] \leftarrow b[m] - a_{m,l} b[l]$
$b[l] \leftarrow \frac{b[l]}{a_{l,l}}$
if $i < P$ then
send $b[l]$ to $p_{\frac{n(i+1)}{P}+1,D_{RC}}$

Running Time : $(\frac{n}{P})^2 + \frac{n(P-1)}{P} max(\frac{2n}{P}+o+2, g) + (3P-4)o + (P-1)g + \frac{P-1}{P}n + P - 2 = O(ng + \frac{n^2}{P})$

Algorithm $A_{D_{CC}}$ and Communication Pattern $C_{D_{CC}}$ for Column Contiguous Data Layout

Every processor $p_{\frac{ni}{P}+1,D_{CC}}$ do in parallel :
for $l = 1$ to i do
for $m = 1$ to $\frac{n}{P}$ do
receive $temp[\frac{ni}{P}+m]$ from $p_{\frac{n(l-1)}{P}+1,D_{CC}}$
$b[\frac{ni}{P}+m] \leftarrow b[\frac{ni}{P}+m] - temp[\frac{ni}{P}+m]$
for $l = 1$ to $\frac{n}{P}$ do
for $m = 1$ to $l-1$ do
$b[\frac{ni}{P}+l] \leftarrow b[\frac{ni}{P}+l] - a_{\frac{ni}{P}+l,m} b[m]$
$b[\frac{ni}{P}+l] \leftarrow \frac{b[\frac{ni}{P}+l]}{a_{\frac{ni}{P}+l,\frac{ni}{P}+l}}$
for $l = \frac{n(i+1)}{P}+1$ to n do

$temp[l] \leftarrow a_{l,\frac{ni}{\bar{P}}+1} b[\frac{ni}{\bar{P}}+1]$
for $m = 2$ to $\frac{n}{\bar{P}}$ do
$temp[l] \leftarrow temp[l] + a_{l,\frac{ni}{\bar{P}}+m} b[\frac{ni}{\bar{P}}+m]$
send $temp[l]$ to $p_{l,D_{CC}}$

Running Time : $\frac{n^2}{\bar{P}} + \frac{n(\bar{P}-1)}{\bar{P}} max(\frac{2n}{\bar{P}}+o, g) + (2\bar{P}-2)o + (\bar{P}-1)g + \bar{P} - 1 = O(ng + \frac{n^2}{\bar{P}})$

For block data layout assume that the processors are labeled as $p_{i,j}$ where $j \leq i \leq \bar{P}$. Processor $p_{i,j}$ is the processor which is initially assigned items $a_{m,l}$ where $\frac{n(i-1)}{\bar{P}} < m \leq \frac{ni}{\bar{P}}$ and $\frac{n(j-1)}{\bar{P}} < l \leq \frac{nj}{\bar{P}}$.

We present two possible sub-algorithms $A^1_{D_{BD}}, A^2_{D_{BD}}$ and communication patterns $C^1_{D_{BD}}, C^2_{D_{BD}}$ for block data layout. The first is the straightforward algorithm. The second redistributes the data into the row contiguous layout and calls the row-contiguous algorithm $A_{D_{RC}}$ with communication pattern $C_{D_{RC}}$. For the second sub-algorithm the processor mapping between the two layouts is as follows : $p_{i,j} = p_{(\frac{i(i-1)}{2}+j-1)\frac{n}{\bar{P}}+1, D_{RC}}$.

Subalgorithm $A^1_{D_{BD}}$ and Communication Pattern $C^1_{D_{BD}}$

Every processor $p_{i,j}$ do in parallel :
if $j = i$ then
for $l = 1$ to $i - 1$ do
for $m = 1$ to $\frac{n}{\bar{P}}$ do
receive $temp[\frac{n(i-1)}{\bar{P}}+m]$ from $p_{j,l}$
$b[\frac{n(i-1)}{\bar{P}}+m] \leftarrow b[\frac{n(i-1)}{\bar{P}}+m] - temp[\frac{n(i-1)}{\bar{P}}+m]$
for l =1 to $\frac{n}{\bar{P}}$ do
for $s = 1$ to $l - 1$ do
$b[\frac{n(i-1)}{\bar{P}}+l] \leftarrow b[\frac{n(i-1)}{\bar{P}}+l] - a_{\frac{n(i-1)}{\bar{P}}+l, \frac{n(i-1)}{\bar{P}}+s}\, b[\frac{n(i-1)}{\bar{P}}+s]$
$b[\frac{n(i-1)}{\bar{P}}+l] \leftarrow \frac{b[\frac{n(i-1)}{\bar{P}}+l]}{a_{\frac{n(i-1)}{\bar{P}}+l, \frac{n(i-1)}{\bar{P}}+l}}$
if $i \neq \bar{P}$ then
send $b[\frac{n(i-1)}{\bar{P}}+l]$ to $p_{i+1,i}$
else
for $l = \frac{n(j-1)}{\bar{P}}+1$ to $\frac{nj}{\bar{P}}$ do
receive $b[l]$ from $p_{i-1,j}$
if $i \neq \bar{P}$ then
send $b[l]$ to $p_{i+1,j}$
for $m = \frac{n(j-1)}{\bar{P}}+1$ to $\frac{nj}{\bar{P}}$ do
$temp[m] \leftarrow a_{m,\frac{n(j-1)}{\bar{P}}+1} b[\frac{n(j-1)}{\bar{P}}+1]$
for $l = \frac{n(j-1)}{\bar{P}}+2$ to $\frac{nj}{\bar{P}}$ do
$temp[m] \leftarrow temp[m] + a_{m,l} b[l]$
send $temp[m]$ to $p_{i,i}$

Running Time : $(2\bar{P}-1)\frac{n}{\bar{P}} max(\frac{2n}{\bar{P}}+o, g) + (2\bar{P}-3)g + (4\bar{P}-3)o + \bar{P} - 2 = O(ng + \frac{n^2}{\sqrt{\bar{P}}})$

Subalgorithm $A^2_{D_{BD}}$ and Communication Pattern $C^2_{D_{BD}}$

Every processor $p_{i,j}$ do in parallel :
if $\bar{P}$ even then

for $l = 1$ to $\frac{n}{2P}$ do
  for $m = 1$ to $\frac{n}{P}$ do
    if i odd **then**
      receive $a_{\frac{in}{P}+l,\frac{(j-1)n}{P}+m}$ from $p_{i+1,j}$
    else
      send $a_{\frac{(i-1)n}{P}+l,\frac{(j-1)n}{P}+m}$ to $p_{i-1,j}$

/* For the next steps, sending and receiving are done simultaneously */
/* $t-1 \equiv [(\frac{i(i-1)}{2}+j) \bmod (\bar{P}+1)] \bmod \lceil \frac{\bar{P}+1}{2} \rceil$ */

for $l = j$ to $\lfloor \frac{\bar{P}+1+i}{2} \rfloor - \lfloor \frac{i}{2} \rfloor$ do
  for $m = 1$ to $\frac{n}{P}$ do
    for $s = 1$ to $\frac{n}{P}$ do
      send $a_{\frac{(i-1)n}{P}+\frac{(l-1)n}{P}+m,\frac{(j-1)n}{P}+s}$
        to $p_{\frac{(i-1)n}{P}+\frac{(l-1)n}{P}+m,D_{RC}}$

for $l = 1$ to $min(j-1, \lfloor \frac{\bar{P}+1+i}{2} \rfloor - \lfloor \frac{i}{2} \rfloor)$ do
  for $m = 1$ to $\frac{n}{P}$ do
    for $s = 1$ to $\frac{n}{P}$ do
      send $a_{\frac{(i-1)n}{P}+\frac{(l-1)n}{P}+m,\frac{(j-1)n}{P}+s}$
        to $p_{\frac{(i-1)n}{P}+\frac{(l-1)n}{P}+m,D_{RC}}$

for $l = t$ to $\frac{i(i-1)+2j}{4(\bar{P}+1)}$ do
  for $m = 1$ to $\frac{n}{P}$ do
    for $s = 1$ to $\frac{n}{P}$ do
      receive $a_{(\frac{i(i-1)}{2}+j-1)\frac{n}{P}+m,\frac{(l-1)n}{P}+s}$
        from $p_{\lfloor \frac{(\frac{i(i-1)}{2}+j-1)\frac{n}{P}+m}{\frac{n}{P}} \rfloor,l}$

for $l =$ 1to $t-1$ do
  for $m = 1$ to $\frac{n}{P}$ do
    for $s = 1$ to $\frac{n}{P}$ do
      receive $a_{(\frac{i(i-1)}{2}+j-1)\frac{n}{P}+m,\frac{(l-1)n}{P}+s}$
        from $p_{\lfloor \frac{(\frac{i(i-1)}{2}+j-1)\frac{n}{P}+m}{\frac{n}{P}} \rfloor,l}$

/* End of send/receive phase */
Call $A_{D_{RC}}$ and $C_{D_{RC}}$

Running time : $\frac{n^2}{P^P}g + \frac{n^2}{P^2} + \frac{n^2}{P^2}g + \lceil \frac{\bar{P}+1}{2} \rceil \frac{n^2}{P^P} + \frac{n(P-1)}{P} max(\frac{2n}{P}+o+2,g) + 3Po + (P-1)g + \frac{P-1}{P}n + P - 2 = O(ng + \frac{n^2}{P} + \frac{n^2}{P^2}g)$

Algorithm $A_{D_{BD}}$ and Communication Pattern $C_{D_{BD}}$ for Block Data Layout
if $\sqrt{P} < g$ then
  Call $A^1_{D_{BD}}$ and $C^1_{D_{BD}}$
else
  Call $A^2_{D_{BD}}$ and $C^2_{D_{BD}}$

Running time : $O(ng + \frac{n^2}{P} + min(\frac{n^2}{\sqrt{P}} + \frac{n^2}{P^2}g))$ (for precise values, see the sub-algorithms)

For block-cyclic data layout assume that the processors are labeled as $p^K_{i,j}$ where $i \leq \sqrt{P}$ and $j \leq \sqrt{P}$. Processor $p^K_{i,j}$ is the processor which is initially assigned items $a_{m,l}$ where $\frac{n(i-1)}{K\sqrt{P}} < m \leq \frac{ni}{K\sqrt{P}}$ and $\frac{n(j-1)}{K\sqrt{P}} < l \leq \frac{nj}{K\sqrt{P}}$.

We present two possible sub-algorithms $A^2_{D_{K,BC}}, A^2_{D_{K,BC}}$ and communication patterns $C^1_{D_{K,BC}}$, $C^2_{D_{K,BC}}$ for block-cyclic data layout. The first is the straightforward algorithm. The second partitions the matrix into two sections. The first section uses the straightforward method and the second section is solved by redistributing the data so that for every $\frac{n}{K}$ rows, every processor receives $\frac{n}{KP}$ contiguous rows.

Subalgorithm $A^1_{D_{K,BC}}$ and Communication Pattern $C^1_{D_{K,BC}}$

Every processor $p_{i,j}^K$ do in parallel :

for $l = 0$ to $K-1$ do

if $i \neq j$ then

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

receive $b[\frac{ln}{K} + \frac{n(j-1)}{K\sqrt{P}} + m]$ from $p_{i-1,j}^K$ /* if $i = 1$ then receive from $p_{\sqrt{P},j}^K$ */

if ($j > 1$ and $i \neq j$) or ($j = 1$ and $i \neq \sqrt{P}$) then

send $b[\frac{ln}{K} + \frac{n(j-1)}{K\sqrt{P}} + m]$ to $p_{i+1,j}^K$ /* if $i = \sqrt{P}$ then send to $p_{1,j}^K$ */

for $s = l$ to $K-1$

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

for $r = 1$ to $\frac{n}{K\sqrt{P}}$ do

if $l = 1$ and $r = 1$ then

$temp[\frac{ns}{K} + \frac{n(i-1)}{K\sqrt{P}} + m] \leftarrow a_{\frac{ns}{K}+\frac{n(i-1)}{K\sqrt{P}}+m, \frac{ns}{k}+\frac{n(j-1)}{K\sqrt{P}}+r} b[\frac{ns}{K} + \frac{n(j-1)}{K\sqrt{P}} + r]$

else

$temp[\frac{ns}{K} + \frac{n(i-1)}{K\sqrt{P}} + m] \leftarrow temp[\frac{ns}{K} + \frac{n(i-1)}{K\sqrt{P}} + m] +$

$a_{\frac{ns}{K}+\frac{n(i-1)}{K\sqrt{P}}+m, \frac{ns}{k}+\frac{n(j-1)}{K\sqrt{P}}+r} b[\frac{n(s-1)}{K} + \frac{n(j-1)}{K\sqrt{P}} + r]$

send $temp[\frac{nl}{K} + \frac{n(i-1)}{K\sqrt{P}} + m]$ to $p_{i,i}^K$

if $i = j$ then

if $l \neq 0$ then

for $m = i+1$ to $\sqrt{P}$ do

for $s = 1$ to $\frac{n}{K\sqrt{P}}$ do

receive $temp[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s]$ from $p_{i,m}^K$

$b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s] \leftarrow b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s] - temp[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s]$

if $l \neq 0$ and $i \neq 1$ then

for $m = 1$ to $i-1$ do

for $s = 1$ to $\frac{n}{K\sqrt{P}}$ do

receive $temp[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s]$ from $p_{i,m}^K$

$b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s] \leftarrow b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s] - temp[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + s]$

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

for $s = 1$ to $m-1$ do

$b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + m] \leftarrow b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + m] -$

$a_{\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m, \frac{ln}{K}+\frac{(j-1)n}{K\sqrt{P}}+s} b[\frac{ln}{K} + \frac{n(j-1)}{K\sqrt{P}} + s]$

$b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + m] \leftarrow \frac{b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m]}{a_{\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m, \frac{ln}{K}+\frac{(i-1)n}{K\sqrt{P}}+m}}$

send $b[\frac{ln}{K} + \frac{n(i-1)}{K\sqrt{P}} + m]$ to $p_{i+1,j}^K$ /*if $i = \sqrt{P}$ then send to $p_{1,j}^K$*/

for $s = l+1$ to $k-1$ do

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

for $r = 1$ to $\frac{n}{K\sqrt{P}}$ do

$b[\frac{sn}{K} + \frac{(i-1)n}{K\sqrt{P}} + m] \leftarrow a_{\frac{sn}{K}+\frac{(i-1)n}{K\sqrt{P}}+m, \frac{ln}{K}+\frac{(j-1)n}{K\sqrt{P}}+r} b[\frac{ln}{K} + \frac{(j-1)n}{K\sqrt{P}} + r]$

Running Time : $2K[(2\sqrt{P}-1)\frac{n}{K\sqrt{P}}max(\frac{2n}{K\sqrt{P}}+o,g)+(2\sqrt{P}-3)g+(4\sqrt{P}-3)o+\sqrt{P}-2]+$
$\sum_{i=1}^{K-1} max(2(K-i)\frac{n^2}{K^2P}+\frac{n}{K\sqrt{P}}o,\frac{n}{K\sqrt{P}}g) = O(ng+\frac{n^2}{K\sqrt{P}}+\frac{n^2}{P})$

Subalgorithm $A^2_{D_{K,BC}}$ **and Communication Pattern** $C^2_{D_{K,BC}}$

Every processor $p^K_{i,j}$ do in parallel :

for $l = 0$ to $\lfloor\frac{k^2}{\sqrt{P}}\rfloor - 1$ do

if $i \neq j$ then

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

receive $b[\frac{ln}{K}+\frac{n(j-1)}{K\sqrt{P}}+m]$ from $p^K_{i-1,j}$ /* if $i = 1$ then receive from $p^K_{\sqrt{P},j}$ */

if ($j > 1$ and $i \neq j$) or ($j = 1$ and $i \neq \sqrt{P}$) then

send $b[\frac{ln}{K}+\frac{n(j-1)}{K\sqrt{P}}+m]$ to $p^K_{i+1,j}$ /* if $i = \sqrt{P}$ then send to $p^K_{1,j}$ */

for $s = l$ to $K-1$

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

for $r = 1$ to $\frac{n}{K\sqrt{P}}$ do

if $l = 1$ and $r = 1$ then

$temp[\frac{ns}{K}+\frac{n(i-1)}{K\sqrt{P}}+m] \leftarrow a_{\frac{ns}{K}+\frac{n(i-1)}{K\sqrt{P}}+m,\frac{ns}{k}+\frac{n(j-1)}{K\sqrt{P}}+r} b[\frac{ns}{K}+\frac{n(j-1)}{K\sqrt{P}}+r]$

else

$temp[\frac{ns}{K}+\frac{n(i-1)}{K\sqrt{P}}+m] \leftarrow temp[\frac{ns}{K}+\frac{n(i-1)}{K\sqrt{P}}+m]+$
$a_{\frac{ns}{K}+\frac{n(i-1)}{K\sqrt{P}}+m,\frac{ns}{k}+\frac{n(j-1)}{K\sqrt{P}}+r} b[\frac{n(s-1)}{K}+\frac{n(j-1)}{K\sqrt{P}}+r]$

send $temp[\frac{nl}{K}+\frac{n(i-1)}{K\sqrt{P}}+m]$ to $p^K_{i,i}$

if $i = j$ then

if $l \neq 0$ then

for $m = i+1$ to $\sqrt{P}$ do

for $s = 1$ to $\frac{n}{K\sqrt{P}}$ do

receive $temp[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s]$ from $p^K_{i,m}$

$b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s] \leftarrow b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s] - temp[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s]$

if $l \neq 0$ and $i \neq 1$ then

for $m = 1$ to $i-1$ do

for $s = 1$ to $\frac{n}{K\sqrt{P}}$ do

receive $temp[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s]$ from $p^K_{i,m}$

$b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s] \leftarrow b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s] - temp[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+s]$

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

for $s = 1$ to $m-1$ do

$b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m] \leftarrow b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m]-$
$a_{\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m,\frac{ln}{K}+\frac{(j-1)n}{K\sqrt{P}}+s} b[\frac{ln}{K}+\frac{n(j-1)}{K\sqrt{P}}+s]$

$b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m] \leftarrow \frac{b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m]}{a_{\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m,\frac{ln}{K}+\frac{(i-1)n}{K\sqrt{P}}+m}}$

send $b[\frac{ln}{K}+\frac{n(i-1)}{K\sqrt{P}}+m]$ to $p^K_{i+1,j}$ /*if $i = \sqrt{P}$ then send to $p^K_{1,j}$*/

for $s = l+1$ to $k-1$ do

for $m = 1$ to $\frac{n}{K\sqrt{P}}$ do

for $r = 1$ to $\frac{n}{K\sqrt{P}}$ **do**

$b[\frac{sn}{K} + \frac{(i-1)n}{K\sqrt{P}} + m] \leftarrow a_{\frac{sn}{K}+\frac{(i-1)n}{K\sqrt{P}}+m, \frac{ln}{K}+\frac{(j-1)n}{K\sqrt{P}}+r} b[\frac{ln}{K} + \frac{(j-1)n}{K\sqrt{P}} + r]$

if $i = j$ **then**

for $m = i+1$ to $\sqrt{P}$ **do**

for $s = 1$ to $\frac{n}{K\sqrt{P}}$ **do**

receive $temp[\frac{\lfloor\frac{K^2}{\sqrt{P}}\rfloor n}{K} + \frac{n(i-1)}{K\sqrt{P}} + s]$ from $p^K_{i,m}$

$b[\frac{\lfloor\frac{K^2}{\sqrt{P}}\rfloor n}{K} + \frac{n(i-1)}{K\sqrt{P}} + s] \leftarrow b[\frac{\lfloor\frac{K^2}{\sqrt{P}}\rfloor n}{K} + \frac{n(i-1)}{K\sqrt{P}} + s] - temp[\frac{\lfloor\frac{K^2}{\sqrt{P}}\rfloor n}{K} + \frac{n(i-1)}{K\sqrt{P}} + s]$

/* For the next steps, sending and receiving are done simultaneously*/

for $l = \lfloor\frac{K^2}{\sqrt{P}}\rfloor$ to $K-1$ **do**

for $m = 0$ to $l - \lfloor\frac{K^2}{\sqrt{P}}\rfloor$ **do**

for $s = i$ to $\sqrt{P}-1$ **do**

for $r = 1$ to $\frac{n}{KP}$ **do**

for $t = 1$ to $\frac{n}{K\sqrt{P}}$ **do**

send $a_{\frac{ln}{K}+\frac{(i-1)n}{K\sqrt{P}}+\frac{sn}{KP}+r, \frac{(l+m)n}{K}+\frac{(j-1)n}{K\sqrt{P}}+t}$ to $p^K_{i,s+1}$

for $s = 0$ to $i-2$ **do**

for $r = 1$ to $\frac{n}{KP}$ **do**

for $t = 1$ to $\frac{n}{K\sqrt{P}}$ **do**

send $a_{\frac{ln}{K}+\frac{(i-1)n}{K\sqrt{P}}+\frac{sn}{KP}+r, \frac{(l+m)n}{K}+\frac{(j-1)n}{K\sqrt{P}}+t}$ to $p^K_{i,s+1}$

for $l = \lfloor\frac{K^2}{\sqrt{P}}\rfloor$ to $K-1$ **do**

for $m = 0$ to $l - \lfloor\frac{K^2}{\sqrt{P}}\rfloor$ **do**

for $s = i$ to $\sqrt{P}-1$ **do**

for $r = 1$ to $\frac{n}{KP}$ **do**

for $t = 1$ to $\frac{n}{K\sqrt{P}}$ **do**

receive $a_{\frac{ln}{K}+\frac{(i-1)n}{K\sqrt{P}}+\frac{(i-1)n}{KP}+r, \frac{(l+m)n}{K}+\frac{sn}{K\sqrt{P}}+t}$ from $p^K_{i,s+1}$

for $s = 0$ to $i-2$ **do**

for $r = 1$ to $\frac{n}{KP}$ **do**

for $t = 1$ to $\frac{n}{K\sqrt{P}}$ **do**

receive $a_{\frac{ln}{K}+\frac{(i-1)n}{K\sqrt{P}}+\frac{(i-1)n}{KP}+r, \frac{(l+m)n}{K}+\frac{sn}{K\sqrt{P}}+t}$ from $p^K_{i,s+1}$

/* End of send/receive phase*/

for $l = \lfloor\frac{K^2}{\sqrt{P}}\rfloor$ to $K-1$ **do**

for $m = 1$ to $\frac{n((i-1)\sqrt{P}+j-1)}{K\sqrt{P}}$ **do**

receive $b[\frac{ln}{K}+m]$ from $p^K_{i,j-1}$ /*if $j = 1$ then receive from $p^K_{i-1,\sqrt{P}}$*/

if $i \neq \sqrt{P}$ and $j \neq \sqrt{P}$ **then**

send $b[\frac{ln}{K}+m]$ send to $p^K_{i,j+1}$ /*if $j = \sqrt{P}$ then send to $p^K_{i+1,1}$*/

for $m = 1$ to $\frac{n}{KP}$ **do**

for $s = 1$ to $m-1$ **do**

$b[\frac{ln}{K} + \frac{n((i-1)\sqrt{P}+j-1)}{KP} + m] \leftarrow b[\frac{ln}{K} + \frac{n((i-1)\sqrt{P}+j-1)}{KP} + m] -$

$a_{\frac{ln}{K}+\frac{n((i-1)\sqrt{P}+j-1)}{KP}+m, \frac{ln}{K}+\frac{n((i-1)\sqrt{P}+j-1)}{KP}+s} b[\frac{ln}{K} + \frac{n((i-1)\sqrt{P}+j-1)}{KP} + s]$

$b[\frac{ln}{K} + \frac{n((i-1)\sqrt{P}+j-1)}{KP} + m] \leftarrow \frac{b[\frac{ln}{K}+\frac{n((i-1)\sqrt{P}+j-1)}{KP}+m]}{a_{\frac{ln}{K}+\frac{n((i-1)\sqrt{P}+j-1)}{KP}+m, \frac{ln}{K}+\frac{n((i-1)\sqrt{P}+j-1)}{KP}+m}}$

send $b[\frac{ln}{K} + \frac{n((i-1)\sqrt{P}+j-1)}{KP} + m]$ to $p^K_{i,j+1}$ /* if $j = \sqrt{P}$ then send to $p^K_{i+1,1}$*/

for $m = \frac{n((i-1)\sqrt{P}+j)}{KP} + 1$ to $\frac{n}{K}$ **do**

receive $b[\frac{ln}{K}+m]$ from $p^K_{i,j-1}$ /*if $j = 1$ then receive from $p^K_{i-1,\sqrt{P}}$*/

if $i \neq \sqrt{P}$ and $j \neq \sqrt{P}$ **then**

send $b[\frac{ln}{K}+m]$ to $p^K_{i,j+1}$ /*if $j = \sqrt{P}$ then send to $p^K_{i+1,1}$*/

for $m = l+1$ to $K-1$ **do**

for $s = 1$ to $\frac{n}{KP}$ do
  for $r = 1$ to $\frac{n}{K}$ do
    $b[\frac{nm}{K} + s] \leftarrow b[\frac{nm}{K} + s] - a_{\frac{nm}{K}+s,\frac{nl}{K}+r} b[\frac{nl}{K} + r]$

Running Time : $\lfloor \frac{K^2}{\sqrt{P}} \rfloor (4\sqrt{P} - 2)[\frac{n}{K\sqrt{P}} max(\frac{2n}{K\sqrt{P}} + o, g) + (2\sqrt{P} - 3)g + (4\sqrt{P} - 3)o + \sqrt{P} - 2] + (K - \lfloor \frac{K^2}{\sqrt{P}} \rfloor)[\frac{n}{K\sqrt{P}} g + \frac{n^2}{K^2 P^{\frac{3}{2}}} + \frac{n}{KP} max(\frac{2n}{K\sqrt{P}} + o + 2, g) + 3(\sqrt{P} - 4)o + (\sqrt{P} - 1)g + \sqrt{P} - 2 + \frac{n^2}{2K^2 P}(K - \lfloor \frac{K^2}{P} \rfloor + 1)g] + \sum_{i=1}^{\lfloor \frac{K^2}{\sqrt{P}} \rfloor} max[2(K - i)\frac{n^2}{K^2 P} + \frac{n}{K\sqrt{P}} o, \frac{n}{K\sqrt{P}} g] + \sum_{i=\lfloor \frac{K^2}{\sqrt{P}} \rfloor}^{K-1} [2(K - i)\frac{n^2}{K^2 P^{\frac{3}{2}}} + \frac{n}{KP} o, \frac{n}{KP} g]$

The running time is $O(ng + \frac{n^2}{P} + \frac{n}{KP} g)$ when $K \leq \sqrt{P}$.

Algorithm $A_{D_{K,BC}}$ and Communication Pattern $C_{D_{K,BC}}$ for Block Cyclic Data Layout
if $K > \sqrt{P}$ and $g > \sqrt{P}$ then
  Call $A^1_{D_{K,BC}}$ and $C^1_{D_{K,BC}}$
else
  Call $A^2_{D_{K,BC}}$ and $C^2_{D_{K,BC}}$
Running time : $O(ng + \frac{n^2}{P} + min(\frac{n^2}{KP} g, \frac{n^2}{K\sqrt{P}}))$ (for precise values, see the sub-algorithms)

For $\bar{D}$, the most favorable data layout, we consider two cases. If $n \leq g$, we call sub-algorithm $A^1_{\bar{D}}$ which is the straightforward uniprocessor algorithm. Else, we call $A_{D_{RC}}$ and $C_{D_{RC}}$.

Subalgorithm $A^1_{\bar{D}}$ (no communication pattern needed)
  for $i = 1$ to n do
    for $j = 1$ to $i - 1$ do
      $b[i] \leftarrow b[i] - a_{i,j} b[j]$
    $b[i] \leftarrow \frac{b[i]}{a_{i,i}}$
Running Time : n^2

Algorithm $A_{\bar{D}}$ and Communication Pattern $C_{\bar{D}}$ for $\bar{D}$
if $n \leq g$ then
  Call $A^1_{\bar{D}}$
else
  Call $A_{D_{RC}}$ and $C_{D_{RC}}$
Running Time : n^2 if $n \leq g$
$(\frac{n}{P})^2 + \frac{n(P-1)}{P} max(\frac{2n}{P} + o + 2, g) + (3P - 4)o + (P - 1)g + \frac{P-1}{P} n + P - 2 = O(ng + \frac{n^2}{P})$ if $n > g$

Clearly, since $o \leq g$, all of these running times are within a constant factor of the corresponding lower bounds. Furthermore, since the algorithms are variants of standard algorithms using simple communication patterns, it follows that employing sophisticated techniques will not yield a significant improvement in the running time.

Another point worth mentioning is that block layouts with $P < min(n, \frac{n^2}{g^2})$ and block-cyclic layouts with $K < min(\sqrt{P}, \frac{n}{P}, \frac{n}{g\sqrt{P}}, g)$ incur much higher running times than the other layouts. In general, these two data layouts are considered for this problem because they are standard layouts for LU decomposition [3]. Therefore it may be possible to mask the extra running times by the

running time needed for LU decomposition. However, if one simply needs to solve $T\mathbf{x} = \mathbf{b}$, it will probably be better to employ one of the other data layouts or to choose appropriate values of P and K if using either block decomposition or block cyclic.

4.6 A note about vector b

In this section we have been assuming that the items of $\mathbf{b}$ are available to every processor. Under this assumption, we obtained tight asymptotic bounds on the running time.

Suppose that the items of $\mathbf{b}$ (i.e. $b_1, b_2, \cdots b_n$) are not initially available to all processors. We can transmit these items to all processors in $O(ng)$ time. One way to achieve this is by (1) sending all the items to some processor p and then (2) have p broadcast all these items to the remaining $P-1$ processors. Clearly it takes at most ng time steps for p to receive all the items of b. Step (2) is called the n-item broadcast problem has been discussed in [11]. In [11], the running time of the n-broadcast problem has been shown to be $O(ng)$. Therefore we can re-distribute the items of b so that every processor has a copy of these items in $O(ng)$ time.

The only bounds in Section 4.1- 4.5 which were smaller than ng were for $T_{\mathcal{A}}(x_n)$ when $n \leq g$ which relied on a uniprocessor algorithm. Therefore, the running times of the applicable algorithms increase by at most constant factors.

5 Banded Triangular Linear Systems

In this section, we generalize the lower bounds presented in the previous subsections from lower bounds for solving triangular systems to lower bounds for solving banded triangular systems. We again consider only substitution algorithms.

The New Problem : Given $T\mathbf{x} = \mathbf{b}$ solve for $\mathbf{x}$, where $T = (a_{i,j})$ is a k-banded (lower) $n \times n$ triangular matrix, $\mathbf{b} = (b_j)$ is a vector of size n, and $\mathbf{x} = (x_j)$ a vector of size n.

We now generalize some of the previous definitions.

Definition 5.1 *Let D be a single-item data layout on P processors. Let $1 \leq c \leq \frac{P}{2}$. If no processor is assigned more than a $\frac{c}{P}^{th}$ fraction of the items of T then D is a $\frac{c}{P}$-data layout for k-banded triangular matrix T. We refer to the set of $\frac{c}{P}$ - data layouts as $\mathcal{D}(\frac{c}{P}, k)$.*

Definition 5.2 *Let $D \in \mathcal{D}(\frac{c}{P}, k)$. Define $i_{n,k,\frac{c}{P}}$ to be the smallest integer i such that the number of non-zero items in row 1 to row i of T is greater than $\frac{2nk-k^2+k}{2}\frac{c}{P}$.*

It follows immediately that $i_{n,k,\frac{c}{P}} \leq i_{n,\frac{c}{P}} \leq \sqrt{\frac{c}{P}}n$.

Definition 5.3 *Define $\bar{D}_k$ to be the data layout in which every processor is assigned every matrix item in T and every item in $\mathbf{b}$.*

Theorem 5.1 *For any A in the class $\mathcal{A}$,*

$$T_{\mathcal{A}}(x_n) \geq \begin{cases} \lfloor \frac{n}{k} \rfloor k^2 & \text{if } k \leq \lfloor \sqrt{g} \rfloor \\ \lfloor \frac{n}{k} \rfloor \lceil \frac{k^2+4k-3}{4} \rceil & \text{if } \lfloor \sqrt{g} \rfloor \leq k \leq g \\ max(\lfloor \frac{n}{k} \rfloor \lceil \frac{k}{4} \rceil g, \frac{nk}{P}) & \text{if } k \geq g \end{cases}$$

Proof: Assume that the data layout is $\bar{D}_k$. We see that solving a k-banded triangular linear system using substitution is as hard as solving $\lfloor \frac{n}{k} \rfloor$ triangular linear systems of size k (that cannot be overlappingly solved) using substitution. Therefore using Theorem 4.2, we obtain the following bounds :

$$T_{A,\bar{D}_k}(x_n) \geq \begin{cases} \lfloor \frac{n}{k} \rfloor k^2 & \text{if } k \leq \lfloor \sqrt{g} \rfloor \\ \lfloor \frac{n}{k} \rfloor \lceil \frac{k^2+4k-3}{4} \rceil & \text{if } \lfloor \sqrt{g} \rfloor \leq k \leq g \\ max(\lfloor \frac{n}{k} \rfloor \lceil \frac{k}{4} \rceil g, \frac{nk}{P}) & \text{if } k \geq g \end{cases}$$

Since $\bar{D}_k$ is the most favorable layout, $T_{A,\bar{D}_k}(x_n) = T_A(x_n)$. □

Theorem 5.2 *Let $A \in \mathcal{A}$ and $D \in \mathcal{D}(\frac{c}{P}, k)$. If $k \geq g$ then*

$$T_{A,D}(x_n) \geq max((n - \sqrt{\tfrac{c}{p}}n + 1)g, \tfrac{nk}{P}).$$

Proof: Follows using a similar argument as that for Theorem 4.1. □

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 5.1 *If $D \in \mathcal{D}(\frac{c}{p}, k)$, then for any $A \in \mathcal{A}$,*

$$T_{A,D}(x_n) \geq max([(\tfrac{\sqrt{2}-1}{\sqrt{2}})n + 1]g, \tfrac{nk}{P}).$$

We see that for $n > g$, the lower bound is $\Omega(ng + \frac{nk}{P})$.

6 Conclusion

In this paper we considered the problem of solving triangular linear systems on parallel distributed-memory machines using substitution. Working within the *LogP* model [2], we were able to derive tight asymptotic bounds on the execution time for this problem and provided algorithms which achieve these bounds.

Specifically, we proved that for sufficiently large matrices ($n \geq g$), the running time of any substitution algorithm is $\Omega(ng + \frac{n^2}{P})$. When we then restrict attention to data layouts in which the number of data items assigned to a processor is bounded, the lower bound is still $\Omega(ng + \frac{n^2}{P})$. Since these bounds are achievable, this shows that simply restricting the proportion of data items assigned to a processor does not result in a significantly higher complexity than assuming all processors have all the data items.

We also derived bounds for several specific data layouts which are commonly used by algorithm designers for this problem. In particular, we showed that block-cyclic data layouts with $K < min(\sqrt{P}, \frac{n}{P}, \frac{n}{g\sqrt{P}}, g)$ and block data layout with $P < min(n, \frac{n^2}{g^2})$ have much higher complexities than all the other data layouts we considered.

We showed that there exist substitution algorithms for these data layouts whose running times are within constant factors of the corresponding lower bounds. Since all of these were variants of standard algorithms using simple communication patterns, this shows that utilizing sophisticated techniques, and designing communication patterns which greatly minimize communication between processors yield no significant benefits toward the running time of a substitution algorithm.

Lastly, we generalize the problem to k-banded triangular linear systems. We showed that for $k \geq g$, the running time is $\Omega(ng + \frac{nk}{P})$. Therefore, as before, we see that designing sparse communication patterns gives no significant benefits.

All 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 steps later.

References

[1] A. Borodin and I. Munro. The computational complexity of algebraic and numeric problems. *American Elsevier, New York*, 1975.

[2] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, May 1993. Also appears as TR No. UCB/CS/92 713.

[3] J. W. Demmel, M. T. Heath, and H. A. van der Vorst. Parallel numerical linear algebra. Technical Report UCB/CSD 93/703, University of California at Berkeley, 1993.

[4] J. Dongarra, R. van de Geijn, and D. Walker. A look at scalable dense linear algebra libraries. In *Scalable High-Performance Computing Conference*. IEEE Computer Society Press, April 1992.

[5] S. Eisenstat, Heath M., C. Henkel, and C. Romaine. Modified cyclic algorithms for solving triangular systems on distributed memory multi-processors. *SIAM J. Sci. Stat. Comput.*, 1988.

[6] M. T. Heath and C. H. Romaine. Parallel solution of triangular systems on distributed - memory multiprocessors. *SIAM J. Sci. Stat. Comput.*, 1988.

[7] D. Heller. A survey of parallel algorithms in numerical linear algebra. *SIAM J. Numer. Anal.*, 29(4), 1987.

[8] High Performance Fortran Forum. *High Performance Fortran Language Specification, Version 0.4*, 1992.

[9] N. Higham. Stability of parallel triangular system solvers. Technical Report Numerical Analysis Report. 236, University of Manchester, 1993.

[10] R. M. Karp and V. Ramachandran. A survey of parallel algorithms for shared memory machines. Technical Report UCB/CSD 88/408, University of California at Berkeley, 1988.

[11] R. M. Karp, A. Sahay, E. E. Santos, and K. E. Schauser. Optimal Broadcast and Summation on the *LogP* Model. In *Fifth Annaul ACM Symposium on Parallel Algorithms and Architectures*, 1993.

[12] G. Li and T. Coleman. A new method for solving triangular systems on distributed - memory message - passing multiprocessors. In *SIAM J. Sci. Stat. Comput.*, 1989.

[13] C. Romine and J. Ortega. Parallel solution of triangular systems of equations. *Parallel Computing*, 6:109–114, 1988.

[14] A. Sameh and R. Brent. Solving triangular systems on a parallel computer. *SIAM J. Numer. Anal.*, 1977.