
Journal of Mathematical Modelling and Algorithms **2:** 217–234, 2003.

Efficient and Optimal Parallel Algorithms for Cholesky Decomposition

EUNICE E. SANTOS[1] and PEI-YUE CHU[2]
[1]*Department of Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061, U.S.A. e-mail: santos@cs.vt.edu*
[2]*Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA 18015, U.S.A. e-mail: pyc1@lehigh.edu*

Abstract. In this paper, we consider the problem of developing efficient and optimal parallel algorithms for Cholesky decomposition. We design our algorithms based on different data layouts and methods. We thereotically analyze the run-time of each algorithm. In order to determine the optimality of the algorithms designed, we derive theoretical lower bounds on running time based on initial data layout and compare them against the algorithmic run-times. To address portability, we design our algorithms and perform complexity analysis on the LogP model. Lastly, we implement our algorithms and analyze performance data.

Mathematics Subject Classifications (2000): 65Y05, 65Y10, 68W40, 68Q17, 68Q25.

Key words: Cholesky decomposition, parallel complexity, algorithms and complexity, LogP model, numerical linear algebra.

1. Introduction

In the scientific and engineering community, developing algorithms for solving linear systems of equations has been a vigorous and on-going activity. The most common direct method for solving systems of linear equations is Gaussian Elimination. One special case of Gaussian elimination is Cholesky decomposition which factorizes matrices with certain special characteristics. Cholesky decomposition arises in areas such as linear programming and boundary element methods, making it a particularly interesting problem to explore.

Even though many parallel algorithms have been developed during the past decades [1, 3, 6, 8, 12, 13, 16, 18, 20], not many have addressed several key design issues, such as portability, data layout, communication contention, and communication bottlenecks at the same time, thus potentially providing inefficient performance. As such, our approach will deal with synthesizing many important issues and concepts; including: portability, data layout, communication schedule, and task partitioning. In this paper, we expand on earlier research [18]. Since we intend to develop parallel algorithms that can perform predictably and efficiently

across a spectrum of computing systems, we will work within the well known LogP model [4].

As stated, one of our design criteria is data layout. We note that designers have studied the effect of data and task partitioning on performance [2, 5, 7, 9, 12–15, 17]. As such, we will investigate the effect of 1-D and 2-D layouts on running time in our overall run-time complexity analysis.

Another criteria we will consider is different parallel methods based on task partitioning. We will focus on two methods in our design of parallel Cholesky algorithms: fan-in and fan-out using various data layouts. For brevity, full proofs have been omitted and can be found in [19].

We note that our approach, and potentially, our results in Cholesky decomposition can be extended into applications in finite element and finite difference methods [10, 11].

Lastly, we will implement our algorithms in order to gather performance data in which to analyze.

2. LogP Model

In order to design portable and efficient parallel algorithms, it is imperative that we select a model that stresses both realism and portability. As such, we will use the LogP model in our analysis. In this section, we provide a brief description of the model.

LogP [4] describes an abstract configuration of a distributed-memory multiprocessor machine. The processors communicate by point-to-point messages. The model intends to be independent of the structure of the network but specifies the performance characteristics of the interconnection network to deal with issues such as portability.

The main parameters of the model are:

L: Latency or delay incurred in communicating a message containing a word from the source processor to the target one.

o: Overhead, the length of time that a processor is either transmitting or receiving a message. During this time, the processor cannot perform other operations $[o \leqslant g]$.

g: the gap defined as the minimum time interval between consecutive message sending or consecutive message receiving at a processor.

P: number of processor/memory pairs running in parallel.

There is an upper bound for numbers of messages, L/g, to send/receive through the network.

All our algorithms satisfy the conditions of the LogP model. For brevity, we present results in this paper for the special case $L = g$.

3. Design Issues for Cholesky Decomposition Algorithms

THE PROBLEM. Given a matrix $A = (a_{i,j})$ of size $n \times n$, where A is symmetric and positive definite, determine $L = (l_{i,j})$, a lower triangular matrix of size $n \times n$ where $A = LL^{\mathrm{T}}$.

We see that

$$l_{i,j} = \frac{a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k}}{l_{j,j}},$$

$$l_{j,j} = \sqrt{a_{j,j} - \sum_{k=1}^{j-1} (l_{j,k})^2}.$$

In order to derive useful run-time complexity results, we define the types of operations allowed on the data and the partial results that can be formed and operated on.

DEFINITION 3.1. A *partial sum* of $l_{i,j}$ is one that has the following value

$$\begin{cases} \sum_{k=1}^{j-1} b_k l_{i,k} l_{j,k} & \text{if } i \neq j, \\ \sum_{k=1}^{j-1} b_k (l_{j,k})^2 & \text{otherwise,} \end{cases}$$

where $b_k = 1$ or 0, and can only be computed by

(1) the addition of 2 other partial sums of $l_{i,j}$,
(2) the multiplication of some $l_{j,k}$ and $l_{i,k}$, or
(3) the square of some $l_{j,k}$.

The partial sum

$$X_{i,j} = \begin{cases} \sum_{k=1}^{j-1} l_{i,k} l_{j,k} & \text{if } i \neq j, \\ \sum_{k=1}^{j-1} (l_{j,k})^2 & \text{otherwise} \end{cases}$$

is called the *full partial sum*.

Building on the definition of partial sum, we can now define a partial sum update.

DEFINITION 3.2. The *partial sum update* of $l_{i,j}$ is the value $\hat{a}_{i,j} = a_{i,j} - X_{i,j}$, and is computed by directly subtracting $a_{i,j}$ with $X_{i,j}$.

From these two definitions, we can now define allowable partial result operations for Cholesky decomposition.

DEFINITION 3.3. A *valid partial result* for a Cholesky decomposition algorithm is either any matrix item $a_{i,j}$, or is the result from

- computing any partial sum,
- computing any partial sum update,
- for any j, the operation of taking the square root of $\hat{a}_{j,j}$, or
- for any i and j where $i \neq j$, the operation of dividing $\hat{a}_{i,j}$ with $l_{j,j}$.

Item 3 and item 4 are referred to as *final valid partial results.*

From these definitions, it is clear that the lower bound on the number of computations is $\Omega(n^3)$.

In serial Cholesky decomposition, it is common to discuss the various looping/nesting techniques in the algorithm design. Clearly, there are three loops or nests that can be used (for i, j, and k indices). This provides $3! = 6$ different looping structures that are named using the nesting order: ijk, ikj, jik, jki, kij, and kji, respectively. Each of these algorithms have a running time of $O(n^3)$. These structures have been grouped via the first nesting loop [5, 15]. The ijk- and ikj-structures are called *row Cholesky*, jki and jik are called *column Cholesky*, and kji and kij are *submatrix Cholesky*.

Submatrix-Cholesky is referred to as immediate-update since as soon as multipliers ($l_{i,j}$'s) are available, values are updated that use them. Both row- and column-Cholesky are referred to as delayed-update since rows or columns, respectively, are updated only just before they are needed. Submatrix-Cholesky is often referred to as *fan-out*, while column-Cholesky is referred to as *fan-in*. In this paper, we will consider the methods of parallel fan-in and parallel fan-out.

While, it is clear that the serial running time of the looping algorithms is $O(n^3)$ by considering the number of operations performed, many issues must be considered in designing and analyzing parallel Cholesky algorithms. In general, a parallel algorithm is composed of three main items:

(a) data layout,
(b) computational tasks, and
(c) communication schedule.

Run-time can only be analyzed after these three items are specified. A data layout is the initial assignment of the items of matrix A to the P processors. In this paper, we consider several important and often-used data layouts in our design and analysis. These will be discussed in the following subsection. Communication is dependent on the layout and computational tasks since they will specify the data and partial results that need to be sent between processors. Furthermore, communication must satisfy conditions of the LogP model. Communication schedules will be discussed in each algorithm. The lower bounds we will derive in Section 4 will hold regardless of specific communication schedules. The set of computational tasks are those whose operations correctly compute L and only create *valid partial results*. Also, when we discuss parallel algorithms based on methods, we must further restrict computational tasks based on assignment to processors.

Upon close inspection of the definitions for creating valid partial results, we see that the computational lower bound on running time for any algorithm (regardless of data layout and communication schedule) is $\Omega(\frac{n^3}{P} + n)$ since the serial complexity is $\Theta(n^3)$, and there is an inherent sequentiality of $\Omega(n)$.

We now discuss *parallel fan-in* and *parallel fan-out* [5]. We note that much of the existing discussion for fan-out and fan-in algorithm design has been based on using column-oriented data layouts. We intend to design algorithms that utilize a variety of data layouts. Therefore, we have *generalized* the methods in order to account for use of different data distributions. In order to do so, we first provide the following definitions.

REMARK 3.1. The processor initially assigned matrix item $a_{i,j}$ is denoted by $p_{i,j}$.

DEFINITION 3.4. A *partial aggregate* of $l_{i,j}$ on *processor* ρ is a partial sum of the form

$$\gamma_{i,j}^{\rho} = \sum_{k=1}^{j-1} b_k (l_{i,k} l_{j,k}),$$

where

$$b_k = \begin{cases} 1 & \text{if } p_{i,k} = \rho, \\ 0 & \text{otherwise.} \end{cases}$$

A fan-out Cholesky parallel algorithm requires that all partial results for $l_{i,j}$ be computed by $p_{i,j}$. A fan-in algorithm requires that partial results be divided amongst the processors based on assigning partial aggregates as tasks. More formally:

DEFINITION 3.5. An algorithm for Cholesky decomposition is a *fan-out* algorithm on P processors if for all i, and j, any valid partial results for computing $l_{i,j}$ in the algorithm is assigned to $p_{i,j}$.

DEFINITION 3.6. An algorithm for Cholesky decomposition is a *fan-in* algorithm on P processors if

- for all i and j, and any processor ρ, any partial sum for computing $\gamma_{i,j}^{\rho}$ in the algorithm is assigned to processor ρ, and
- $p_{i,j}$ performs the summation operation that computes the value $X_{i,j}$, the partial sum update $\hat{a}_{i,j}$, and the final valid partial result for $l_{i,j}$.

A fan-out algorithm must rely on transmissions of $l_{i,j}$'s between processors in order to perform partial sums. Any other valid partial result sent among processors would be counterproductive since processor $p_{i,j}$ cannot use results from other processors, and $a_{i,j}$ is already initially assigned to $p_{i,j}$.

A fan-in algorithm relies on transmissions of partial sums in order for $p_{i,j}$ to gather the partial sums needed to form $X_{i,j}$.

3.1. DATA LAYOUT

In designing parallel Cholesky algorithms, the choice of an initial data layout has significant ramifications in design. A design not accounting for the layout could lead to significant slowdowns in performance. As such, data layouts are listed as one of the three main factors in our design and analysis. We consider well-known matrix layouts that have been used in parallel Cholesky design. These layouts fall mainly into two group: one-dimensional (1-D) and two-dimensional (2-D). In this section, we provide definitions for the layouts we will use in this paper.

DEFINITION 3.7. Given P ($\leqslant n$) processors, denoted $\rho_0, \rho_1, \ldots, \rho_{p-1}$, and an integer β where $1 \leqslant \beta \leqslant n/P$, a one-dimensional *column wrapped layout* of *panel size* β, is a layout where for all i, $0 \leqslant i < P$, ρ_i is assigned column $k\beta P + i\beta + q$, for all $0 \leqslant k < \frac{n}{\beta P}$ and $1 \leqslant q \leqslant \beta$.

We note that if row-oriented data layouts were to be used, there is no fundamental difference between the fan-in and fan-out general methods. As such, row data layouts are not a focus of this paper.

For two-dimensional data layouts, blocks of the matrix A are assigned across the processors. There is a granularity value G that is used to determine the size of the (square) blocks to distribute.

DEFINITION 3.8. Given P ($\leqslant n^2$) processors, denoted $\rho_{i,j}$ where $1 \leqslant i, j \leqslant \sqrt{P}$, and an integer G where $1 \leqslant G \leqslant n/\sqrt{P}$, a two-dimensional *block cyclic* layout of *granularity size* G assigns to each processor $\rho_{i,j}$ the matrix items $a_{l,m}$ where $\frac{n}{G}k + \frac{n}{G\sqrt{P}}(i-1) + 1 \leqslant l \leqslant \frac{n}{G}k + \frac{n}{G\sqrt{P}}i$, $\frac{n}{G}k + \frac{n}{G\sqrt{P}}(j-1) + 1 \leqslant m \leqslant \frac{n}{G}k + \frac{n}{G\sqrt{P}}j$, and $0 \leqslant k < G$.

When $G = 1$, the layout is also known as a *block decomposition* layout.

4. Lower Bounds on Run-Time

In order to understand and measure the efficiency of a parallel algorithm, it is important to derive a lower bound on running time to use for comparison. The closer an algorithm is to an asymptotic lower bound, the better guarantee of efficiency can be made. As such, in this section we derive a variety of lower bounds on parallel run-time on the LogP model. Our lower bounds assume a particular data layout and method while holding for any correct communication schedule that is used in a Cholesky decomposition algorithm.

We note that since there are $\Theta(n^3)$ operations in order to perform Cholesky decomposition, and an inherent sequentiality of $\Omega(n)$, an inherent lower bound would be $\Omega(\frac{n^3}{P} + n)$. This is referred to as the computational lower bound. Clearly, the computational lower bound is a major factor in run-time, but does not include communication time. In this section, we will consider the effects of communication

on the lower bound and derive results that synthesize communication and computation. We note that in our analyses, we make the realistic assumption that $n \gg g$. For brevity, proofs have been omitted, but can be found in [19].

Our first lower bound holds for any parallel fan-out Cholesky algorithm that utilizes a column-wrapped layout with panel size β. A fan-out algorithm requires that every $p_{i,j}$ perform the full set of computations to compute $l_{i,j}$ with no help from other processors except to receive the needed $l_{i,k}$'s and $l_{j,k}$'s. Consider any column α in the matrix, in order to compute all the $l_{i,\alpha}$'s, it must receive partial columns from the first $\alpha - 1$ columns (if the columns belong to other processors). Clearly, processor ρ_P must receive many $l_{a,b}$'s. Our lower bound on communication is based on analyzing the lower bound on the number of messages ρ_P must receive. Below is the combined (computational and communication) lower bound for fan-out on panel column-wrapped layouts.

THEOREM 4.1. *A lower bound for parallel fan-out Cholesky decomposition on P processors using column-wrapped data layout with panel size β is $\Omega(\frac{n^3}{P} + n^2 g)$.*

Analyzing the lower bound derived, we note that (1) the communication lower bound is $\Omega(n^2 g)$, which pinpoints that when designing an algorithm under these assumptions, there is no need to try to design it to perform (asymptotically) less than n^2 phases of communication. Moreover, β does not seem to have an effect on the asymptotic lower bound.

We now derive a lower bound on run-time for fan-in Cholesky decomposition using a panel column-wrapped layout. Since a column-oriented layout is used, the only useful messages sent would contain either values of partial aggregates or values of sums of partial aggregates. Again, consider any column α, the processor assigned column α must take pairs of $l_{i,\alpha}$ and $l_{j,\alpha}$ for multiplication in order to form partial aggregates. There are $\Omega((n-\alpha)^2)$ such partial aggregates from multiplications from this column alone. The lower bound on communication considers the amount of messages a processor needs to send in order to transmit the needed partial aggregates to appropriate processors. We obtain the following lower bound result:

THEOREM 4.2. *A lower bound for parallel fan-in Cholesky decomposition on P processors using column-wrapped data layout with panel size β is $\Omega(\frac{n^3}{P} + n^2 g)$.*

Interestingly, (a) fan-in and fan-out for one-dimensional column-wrapped data layout with panel size β have asymptotically equal lower bounds, (b) β does not seem to asymptotically affect run-time, and (c) P does not appear to be an asymptotic factor in the communication part of the lower bound.

Our lower bound results above for fan-in and fan-out are fixed on a restricted class of 1-D data column layouts, namely structured wrapped layouts with fixed panel size. We now focus our attention on deriving a lower bound that is applicable to all 1-D column-oriented layouts. Thus, we begin by providing the following definition:

DEFINITION 4.1. A data layout is said to be a *single-item* data layout if every data item is initially assigned to a unique processor.

A single-item layout is one that does not allow multiple copies of the data to be initially distributed among processors. This is a realistic assumption, especially in light of the fact that when designers typically turn to parallel environments, it is because the matrix sizes are quite large making replication of data costly. Another realistic assumption is that processors are assigned an equal portion of columns. Therefore, we now formally define this.

DEFINITION 4.2. A data layout on matrix A of size $n \times n$ for P processors is said to be a *balanced* data layout if every processor is assigned $\theta(\frac{n^2}{P})$ of the matrix items.

Building on this definition, we can discuss balanced column-oriented data layouts.

DEFINITION 4.3. A data layout for matrix A on P processors is said to be a *balanced column* data layout if

- it is a single-item layout,
- it is a balanced layout, and
- any two matrix items $a_{i,j}$ and $a_{k,j}$ that are in the same column must be assigned to the same processor ρ.

Using this data layout, we obtain the following:

THEOREM 4.3. *Given any balanced column data layout,* $\Omega(\frac{n^3}{P} + n^2 g)$ *is a lower bound on running time for both fan-in and fan-out Cholesky decomposition.*

From the theorem above, we see that the balanced column layout lower bounds are asymptotically equal to the lower bounds for panel column-wrapped. Furthermore, we see that if a column-oriented layout is used, as long as there are no initial replications (i.e. the layout is single-item), and data is distributed proportionally across processors (i.e. the layout is balanced), the lower bounds are asymptotically equal between fan-in and fan-out Cholesky decomposition.

As an aside, if we generalize our result such that *any* single-item, balanced, one-dimensional data layout (i.e. either row- or column-assignment), we can obtain a lower bound on run-time for fan-in and fan-out, and the corollary is provided below.

COROLLARY 4.4. *Given any single-item, balanced, one-dimensional data layout,* $\Omega(\frac{n^3}{P} + n^2 g)$ *is a lower bound on running time for both fan-in and fan-out Cholesky decomposition.*

We now turn our attention to deriving lower bounds for two-dimensional data layouts. These layouts have a granularity size G for subdividing the matrix into sub-blocks for assignment to processors.

In fan-out, every $p_{i,j}$ must compute the full set of valid partial results to form $l_{i,j}$. Thus, consider processor $\rho_{s,t}$ in the block cyclic layout. This processor is assigned G^2 sub-blocks each of size $\frac{n^2}{GP}$. Our lower bound is derived by computing the number of messages processor $\rho_{\sqrt{P},\sqrt{P}}$ must receive in order to compute all of it's $l_{i,j}$'s.

THEOREM 4.5. *A lower bound for parallel fan-out Cholesky decomposition on P processors using block cyclic data layout with granularity size G is* $\Omega(\frac{n^3}{P} + \frac{n^2}{\sqrt{P}}g)$.

Analyzing our results, we see that the granularity size G does not appear in the lower bound above.

Turning our attention to fan-in, we again consider processor $\rho_{s,t}$. This processor must compute all partial aggregates $\gamma_{i,j}^{\rho_{s,t}}$. We see that a message can either be a transmission of a partial aggregate value, or a transmission of some $l_{i,j}$ value among two processors. Therefore if processor $\rho_{s,t}$ is assigned matrix item $a_{i,j}$, it must have either computed or received $l_{k,j}$ where $j \leqslant k \leqslant i$ in order to compute it set of partial aggregates. Based on this fact, we obtain the following lower bound for fan-in Cholesky decomposition.

THEOREM 4.6. *A lower bound for parallel fan-in Cholesky decomposition on P processors using block cyclic data layout with granularity size G is* $\Omega(\frac{n^3}{P} + \frac{n^2}{\sqrt{P}}g)$.

Again, the choice of fan-in or fan-out does not seem to affect the lower bound analysis. Also, granularity does not seem to affect the derived lower bounds asymptotically.

Overall, the choice of fan-in versus fan-out does not seem to have a major effect on the lower bounds derived.

The lower bounds derived, clearly, do not guarantee that a parallel algorithm can be designed that can achieve each asymptotic lower bound. The goal is to use these asymptotic values as a means of an absolute comparison in order to determine the efficiency of an algorithm. Of course, we will try to design algorithms that do achieve our lower bounds, if the asymptotic run-time of an algorithm equals an asymptotic lower bound, then that algorithm is asymptotically optimal.

5. Parallel Algorithms

In order to discuss our parallel algorithm designs, we first discuss serial fan-in and serial fan-out. In order to do so, two subroutines called *CMOD* for partial sum update, and *CDIV* for generating Cholesky factors $l_{i,j}$ for column j are first presented. The pseudo-code for the subroutines are written to appropriately address our formal definitions of valid partial results.

ALGORITHM 5.1 $CMOD(r, s)$

for $i = r$ to n do
  if $s = 1$ then
    $\hat{l}_{i,r} \leftarrow l_{i,s} * l_{r,s}$
  else
    $\hat{l}_{i,r} \leftarrow \hat{l}_{i,r} + l_{i,s} * l_{r,s}$

ALGORITHM 5.2 $CDIV(s)$

if $s = 1$ then
  $l_{s,s} \leftarrow \sqrt{a_{s,s}}$
  for $i = s + 1$ to n do
    $l_{i,s} \leftarrow a_{i,s} \div l_{s,s}$
else
  $\hat{l}_{s,s} \leftarrow a_{s,s} - \hat{l}_{s,s}$
  $l_{s,s} \leftarrow \sqrt{\hat{l}_{s,s}}$
  for $i = s + 1$ to n do
    $\hat{l}_{i,s} \leftarrow a_{i,s} - \hat{l}_{i,s}$
    $l_{i,s} \leftarrow \hat{l}_{i,s} \div l_{s,s}$

Fan-in (Column-Cholesky) and fan-out (Submatrix-Cholesky) utilize these subroutines [15] in their design. Below are examples of serial fan-in and serial fan-out.

ALGORITHM 5.3 Submatrix-Cholesky (fan-out)

for $k = 1$ to n do
  $CDIV(k)$
  for $j = k + 1$ to n do
    $CMOD(j, k)$

ALGORITHM 5.4 Column-Cholesky (fan-in)

for $j = 1$ to n do
  for $k = 1$ to $j - 1$ do
    $CMOD(j, k)$
  $CDIV(j)$

In our parallel design, we modify these routines in order to make efficient use of a parallel environment. These modifications also take into account communication costs through use of the LogP model.

For brevity, we present all the running times of our algorithms and present the fan-in and fan-out Cholesky algorithms for panel column-wrapped layout. Full algorithms can be found in [19].

We first consider our one-dimensional data layout, column-wrapped layout with panel size β. Our algorithms are based on existing algorithms for parallel Cholesky

on column layouts [5] that have been adapted to our constraints, and bears much similarity to them.

Our first algorithm is parallel fan-out cholesky for 1-D column wrapped layout.

We start by considering what modifications are needed in *CMOD* and *CDIV* in order to account for initial processor assignment. These new routines will be called $PMOD_{\text{fan-out,1-D}}$ and $PDIV_{\text{fan-out,1-D}}$, respectively.

For fan-out, messages contain only $l_{i,j}$ values. Also, since the data layout is a column-oriented layout, there is no need to modify *CMOD*. Therefore, $PMOD_{\text{fan-out,1-D}}$ is the same algorithm as *CMOD*. However, *CDIV* will be modified since the messages that need to be sent will be partial results created within *CDIV*, so $PDIV_{\text{fan-out,1-D}}$ has been designed such that embedded in it is the communication schedule for the Parallel Fan-out algorithm for column-wrapped. The new *PDIV* algorithm is presented below.

ALGORITHM 5.5 $\text{PDIV}_{\text{fan-out,1-D}}(s, \alpha, b)$

```
if s = 1 then
  if b = α then
    l_{s,s} ← √a_{s,s}
  for i = s + 1 to n do
    if b = α then
      l_{i,s} ← a_{i,s} ÷ l_{s,s}
      send l_{i,s} to processor ρ_{(α+1) mod P}
    else
      receive l_{i,s} from processor ρ_{(α−1) mod P}
      if b ≠ (α + 1) mod P then
        send l_{i,s} to processor ρ_{(α+1)(mod)P}
else
  if b = α then
    l̂_{s,s} ← a_{s,s} − l̂_{s,s}
    l_{s,s} ← √l̂_{s,s}
  for i = s + 1 to n do
    if b = α then
      l̂_{i,s} ← a_{i,s} − l̂_{i,s}
      l_{i,s} ← l̂_{i,s} ÷ l_{s,s}
      send l_{i,s} to processor ρ_{(α+1) mod P}
    else
      receive l_{i,s} from processor ρ_{(α−1) mod P}
      if b ≠ (α + 1) mod P then
        send l_{i,s} to processor ρ_{(α+1)(mod) P}
```

The remainder of the discussion for the parallel fan-out algorithm for panel column-wrapped is straightforward since communication has been detailed in *PDIV*.

It is important that the algorithm not have excess idleness, thus we consider that issue and other issues in our design. Below is the pseudo-code and running time.

ALGORITHM 5.6 Parallel Fan-out Cholesky for Panel Column-Wrapped Layout

For each processor ρ_j do in parallel
  for $k = 0$ to $\frac{n}{\beta P} - 1$ do
   for $s = 0$ to $P - 1$ do
     for $q = 1$ to β do
      $i \leftarrow k\beta P + s\beta + q$
      $PDIV_{\text{fan-out,1-D}}(i, j, s)$
      if $j = s$ then
        for $x = q + 1$ to β do
          $y \leftarrow k\beta P + s\beta + x$
          $PMOD_{\text{fan-out,1-D}}(y, i)$
      for $l = k + 1$ to $\frac{n}{\beta P} - 1$ do
        for $x = 1$ to q do
          $y \leftarrow l\beta P + j\beta + x$
          $PMOD_{\text{fan-out,1-D}}(y, i)$

Run Time: $O(\frac{n^3}{p} + n^2 g)$

Our algorithm running time is asymptotically equal to the corresponding lower bound. As such, this algorithm is asymptotically optimal. Therefore, there is no benefit (beyond reduction of constant factors) to trying to further overlap communication with computation, nor for developing complicated communication schedules.

We now turn our attention towards developing a fan-in Cholesky algorithm for panel column-wrapped data layout. Again, we consider revising *CMOD* and *CDIV* as needed. Messages for this algorithm will be partial sums that contain combinations of partial aggregates. The communication schedule will be embedded inside the two subroutines: $PMOD_{\text{fan-in,1-D}}$ and $PDIV_{\text{fan-in,1-D}}$. We present the pseudo-code for the new *PMOD* and *PDIV* below.

ALGORITHM 5.7 $PMOD_{\text{fan-in,1-D}}(r, s, \alpha, w, m)$

for $i = r$ to n do
  if $s = \alpha\beta + 1$ then
    $\bar{l}^{\alpha}_{i,r} \leftarrow l_{i,s} * l_{r,s}$
  else
    $\bar{l}^{\alpha}_{i,r} \leftarrow \bar{l}^{\alpha}_{i,r} + l_{i,s} * l_{r,s}$
  if $(w + 1)\beta P + \alpha\beta > r$ and $m = \beta$ then
    if $w > 0$ or ($w = 0$ and $\alpha > 1$)
      receive $\bar{l}^{\alpha-1}_{i,r}$ from $\rho_{(\alpha-1) \bmod P}$
      $\bar{l}^{\alpha}_{i,r} \leftarrow \bar{l}^{\alpha}_{i,r} + \bar{l}^{\alpha-1}_{i,r}$
    send $\bar{l}^{\alpha}_{i,r}$ to $\rho_{(\alpha+1) \bmod P}$

ALGORITHM 5.8 $PDIV_{\text{fan-in,1-D}}(s, \alpha)$

```
if s = 1 then
  l_{s,s} ← √a_{s,s}
  for i = s + 1 to n do
    l_{i,s} ← a_{i,s} ÷ l_{s,s}
else if s ⩽ β then
  l̂_{s,s} ← a_{s,s} − l̄^α_{s,s}
  l_{s,s} ← √l̂_{s,s}
  for i = s + 1 to n do
    l̂_{i,s} ← a_{i,s} − l̄^α_{i,s}
    l_{i,s} ← l̂_{i,s} ÷ l_{s,s}
else
  receive l̄^{α−1}_{s,s} from processor ρ_{(α−1) mod P}
    if s = αβ + 1 then
      l̂_{s,s} ← temp_{s,s}
    else
      l̂_{s,s} ← l̄^α_{s,s} + l̄^{α−1}_{s,s}
  l̂_{s,s} ← a_{s,s} − l̂_{s,s}
  l_{s,s} ← √l̂_{s,s}
  for i = s + 1 to n do
    receive l̄^{α−1}_{i,s} from processor ρ_{(α−1) mod P}
      if s = αβ + 1 then
        l̂_{i,s} ← l̄^{α−1}_{i,s}
      else
        l̂_{i,s} ← l̄^α_{i,s} + l̄^{α−1}_{i,s}
    l̂_{i,s} ← a_{i,s} − l̂_{i,s}
    l_{i,s} ← l̂_{i,s} ÷ l_{s,s}
```

Using the modified *PDIV* and *PMOD*, we now design our parallel fan-in Cholesky for column-wrapped layouts.

ALGORITHM 5.9 Parallel Fan-in Cholesky for Panel Column-Wrapped Layout

```
For each processor ρ_j do in parallel
  for k = 0 to n/(βP) − 1 do
    for s = 0 to P − 1 do
      for q = 1 to β do
        l ← kβP + sβ + q
        for t = 0 to k − 1 do
          for x = 1 to β do
            y ← tβP + jβ + x
            PMOD_{fan-in,1-D}(l, y, j, k, x)
```

if $s < j$ then
  for $x = 1$ to β do
    $y \leftarrow k\beta P + j\beta + x$
    $PMOD_{\text{fan-in,1-D}}(l, y, j, k, x)$
else if $s = j$ then
  for $x = 1$ to $q - 1$ do
    $y \leftarrow k\beta P + j\beta + x$
    $PMOD_{\text{fan-in,1-D}}(l, y, j, k, x)$
$PDIV_{\text{fan-in,1-D}}(l, j)$

Run Time: $O(\frac{n^3}{p} + n^2 g)$

Analyzing our run-time result for parallel fan-in on column layouts, and comparing it with the lower bounds derived in the previous section, we see that we have an asymptotically optimal algorithm.

Thus, we have provided asymptotically optimal algorithms for both fan-in and fan-out for column-wrapped layouts. Since the running times also match the lower bounds for our arbitrarily one-dimensional layouts, our algorithms are asymptotically optimal over that range of data layouts. Furthermore, β does not asymptotically affect run-time complexity for 1-D layouts.

Turning our attention to block-cyclic layouts, for brevity, we have omitted the algorithms and provided the running times. Full algorithms can be found in [19].

The running times of our parallel fan-in Cholesky and parallel fan-out Cholesky on block cyclic layouts with granularity size G are the same; namely $O(\frac{n^3}{p} + n^2 g)$. Comparing the running times with the lower bounds derived, clearly the two are not asymptotically equal. We note that the asymptotic running times of the algorithms for block cyclic are the same as those for column-wrapped. Moreover, the column-wrapped algorithms are asymptotically optimal under the restriction of using 1-D layouts. Since our algorithms for 2-D layouts already as efficient as the respective 1-D counterparts and the algorithms for 1-D layouts cannot be made to be asymptotically faster, we are guaranteed that the run-time complexity for parallel fan-in and fan-out Cholesky using 2-D layouts are either equal, or even better than the complexity with 1-D layouts. This spotlights the usefulness in further work towards determining the asymptotic complexity for Cholesky decomposition using these data layouts.

6. Experimental Results

In the previous sections, we have provided a theoretical analysis on the complexity of parallel Cholesky decomposition using fan-in and fan-out methods and on two basic layouts by deriving lower bounds, designed and analyzed algorithms, and compared the various results. In this section, we implement our algorithms and gather performance data in order to compare them against our theoretical predictions.

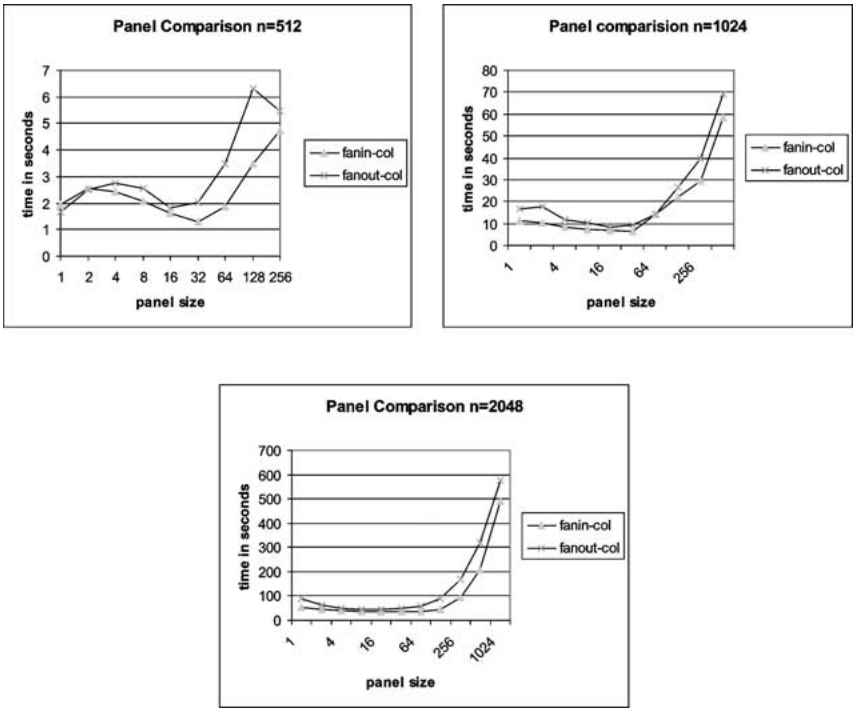

Figure 1. Effects of panel size (β) on performance for $n = 512$, 1024, and 2048.

We implemented our algorithms on a cluster of Pentium III (1 GHz) machines using MPI (Message Passing Interface). In our implementations, we utilized MPIsend and MPIrecv commands as the basic primitives for building our own communication schedules.

We implemented our algorithms for a variety of matrix sizes spanning from $n = 512$, 1024, and 2048 and used from 2 to 16 processors (for block cyclic, only 4 and 16 processors were used due to subdividing using $\sqrt{P}$). For panel column wrapped layouts, the panel size (β) spanned from 1 to 1024 (dependent of the size of the matrix and the number of processors used). For block cyclic layouts, the granularity size (G) spanned from 1 to 1024 (dependent on n and P). For brevity, only key results are presented. Full details can be found in [19].

In our analysis, we noted that the panel size, β, has no asymptotic effect on running time. We now compare the performance of parallel fan-in and fan-out Cholesky decomposition on column-wrapped layouts based on panel size. Figure 1 presents a performance comparison based on panel size for $n = 512$, 1024, and 2048, respectively. Clearly, panel size does have an effect on performance. Therefore, the asymptotic analysis is hiding the effects of β within constant factors.

We next compare the performance of parallel fan-in and parallel fan-out using panel column-wrapped data layouts. Figure 2 presents performance for $n =$

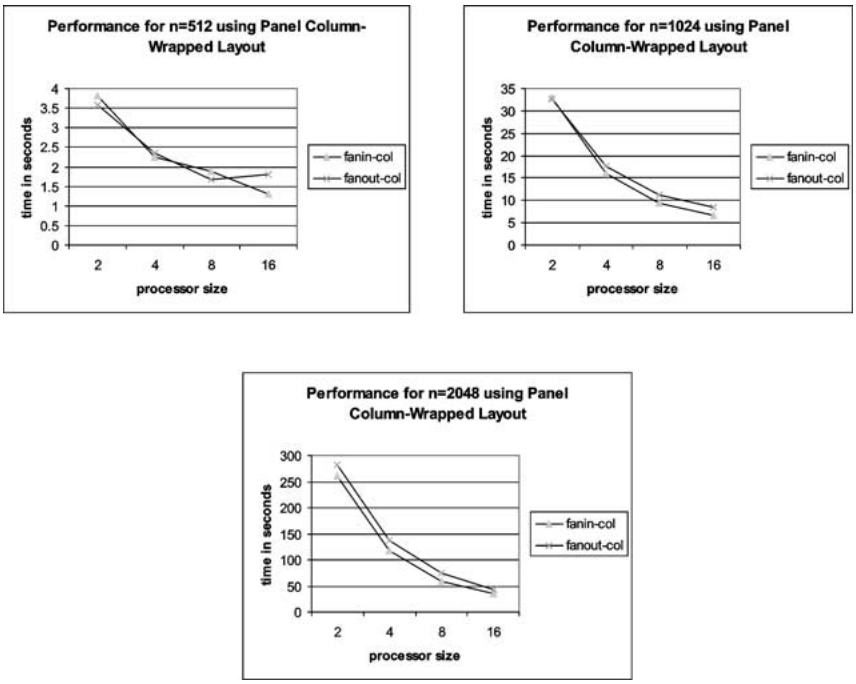

Figure 2. Performance for $n = 512$, 1024, and 2048 of parallel fan-in and fan-out using panel column-wrapped layout.

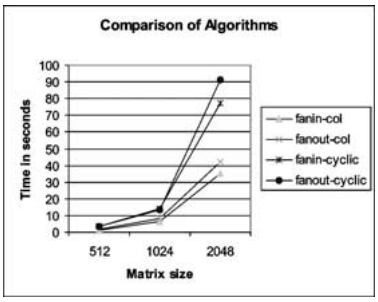

Figure 3. Performance comparison of the algorithms.

512, 1024 and 2048. Analyzing the data, using panel column-wrapped, differences in parallel fan-in and parallel fan-out performance are quite small.

Turning our attention to block cyclic layouts, we note that similar to β in the one-dimensional panel column-wrapped layout, G does affect performance, which seems to be hidden in the asymptotic analysis. Full performance data for block cyclic can be found in [19].

Lastly, Figure 3 provides a comparison of the algorithms implemented. Similar to the 1-D layout algorithms, the 2-D layout algorithms difference in overall performance are not high. However, the 1-D layouts appear to become more efficient

as the problem size increases. This is a topic for future work since a variety of factors could be causing this phenomenon.

7. Conclusion

In this paper, we considered the problem of designing efficient and optimal parallel Cholesky decomposition algorithms for general distributed-memory networks/machines. We considered two standard layouts; namely panel column-wrapped data layout and block cyclic data layout that would be used to design the algorithms. We also considered generalized fan-in and fan-out methods for Cholesky decomposition as the two methods in our analyses.

We derived both upper and lower bounds on run-time in order to determine efficiently and optimality. From our theoretical analysis, we showed that both algorithms using our 1-D layouts are asymptotically optimal. Furthermore, we showed that there is no need to use more complicated 1-D layouts in order to obtain a better running time (asymptotically) since our lower bounds hold for a generic set of 1-D layouts.

For 2-D layouts, we were able to show that our algorithms are efficient but not necessarily optimal, since the upper and lower bounds are not equal. This leaves open the question of whether the lower bound, upper bound, or both need to be tightened.

Furthermore, we implemented our algorithms in order to provide performance analysis. Our analysis was able to pinpoint and predict important performance issues that need to be considered for optimal and/or efficient parallel algorithm design. We note that some performance issues were marked by the asymptotic nature of our analysis suggesting directions for future work.

Acknowledgements

Eunice E. Santos was supported in part by an NSF CAREER Grant.

References

1. Agarwal, R. C., Gustavson, F. G. and Zubair, M.: Improving performance of linear algebra algorithms for dense matrices using algorithmic prefetch, *IBM J. Res. Develop.* **38**(3) (1994), 265–275.
2. Arioli, M., Duff, I. S., Noailles, J. and Ruiz, D.: A block projection method for sparse matrices, *SIAM J. Sci. Statist. Comput.* **13** (1992), 47–70.
3. Ashcraft, C.: The fan-both family of column-based distributed Cholesky factorization algorithms, Technical Report ECA-TR-208, Boeing Computer Services, 1992.
4. Culler, D., Karp, R., Patterson, D., Sahay, A., Santos, E. E., Schauser, K., Subramonian, R. and von Eicken, T.: LogP: A practical model for parallel computation, *Comm. ACM* **37**(11) (1996).
5. Demmel, J. W., Heath, M. T. and van der Vorst, H. A.: Parallel numerical linear algebra, *Acta Numer.* **2** (1993).

6. Dongarra, J. J., Gustavson, F. G. and Karp, A.: Implementing linear algebra algorithms for dense matrices on a vector pipeline machine, *SIAM Rev.* **26** (1986), 91–112.
7. Dumitrescu, B., Doreille, M., Roch, J. L. and Trystram, D.: Two-dimensional block partitioning for the parallel sparse Cholesky factorization, *Numer. Algorithms* **16** (1997), 17–38.
8. Edelman, A.: Large dense numerical linear algebra in 1993: The parallel computing influence, *Int. J. Supercomput. Appl.* **7** (1994).
9. Eswar, K., Huang, C.-H. and Sadayappan, P.: On mapping data and computation for parallel sparse Cholesky factorization, *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, 1995.
10. Gravannis, G. A.: Explicit preconditioned methods for solving 3D boundary value problems by approximate inverse finite element matrix techniques, *Int. J. Comp. Math.* **56** (1995).
11. Gravvanis, G. A.: A three-dimensional symmetric linear equation solver, *Comm. Numer. Methods Engrg.* **10**(9) (1994).
12. Gustavson, F. G.: Recursive leads to automatic variable blocking for dense linear-algebra algorithms, *IBM J. Res. Develop.* **41**(6) (Nov. 1997).
13. Lichtenstein, W. and Johnsson, S. L.: Block-cyclic dense linear algebra, *SIAM J. Sci. Comput.* **14**(6) (1993).
14. Ng, E. G. and Peyton, B. W.: Block sparse Cholesky algorithms on advanced uniprocessor computers, *SIAM J. Sci. Statist. Comput.* **14** (1993), 1034–1056.
15. Ortega, J. M.: *Introduction to Parallel and Vector Solutions of Linear Systems*, Plenum Press, New York, 1988.
16. Rothberg, E. and Gupta, A.: The performance impact of data reuse in parallel dense Cholesky factorization, Technical Report CS-TR-92-1401, Stanford University, Jan. 1992.
17. Rothberg, E. and Gupta, A.: An efficient block-oriented approach to parallel sparse Cholesky factorization, *SIAM J. Sci. Comput.* **15**(6) (1994), 1413–1439.
18. Santos, E. E. and Chu, P.: Efficient parallel algorithms for dense Cholesky factorizition, In: Lecture Notes in Comput. Sci. 1557, Springer, New York, 1999, pp. 600–602.
19. Santos, E. E. and Chu, P.: Optimal and efficient parallel algorithms for dense Cholesky decomposition, Technical Report LCID-03-102, Laboratory for Computation, Information, & Distributed Processing, Virginia Polytechnic Institute & State University, 2003.
20. Strazdins, P. E.: High performance dense linear systems solution on a beowulf cluster, In: *5th International Conference on High Performance Computing in Asia*, 2001.