NASA Contractor Report 181863

ICASE Report No. 89–40

ICASE

DATA TRAFFIC REDUCTION SCHEMES FOR CHOLESKY FACTORIZATION ON ASYNCHRONOUS MULTIPROCESSOR SYSTEMS

Vijay K. Naik
Merrell L. Patrick

.... NAS1-18107, NAS1-18605

June 1989

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665–5225

Operated by the Universities Space Research Association

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

Data Traffic Reduction Schemes for Cholesky Factorization on Asynchronous Multiprocessor Systems*

Vijay K. Naik

IBM Research, T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, N.Y. 12598, U.S.A.

Merrell L. Patrick

ICASE, NASA Langley Research Center
Hampton, VA 23665, U.S.A.
and
Computer Science Department, Duke University
Durham, N.C. 27706, U.S.A.

Abstract: Communication requirements of Cholesky factorization of dense and sparse symmetric, positive definite matrices are analyzed. The communication requirement is characterized by the data traffic generated on multiprocessor systems with local and shared memory. Lower bound proofs are given to show that when the load is uniformly distributed the data traffic associated with factoring an $n \times n$ dense matrix using n^α, $\alpha \leq 2$, processors is $\Omega(n^{2+\alpha/2})$. For an $n \times n$ sparse matrices representing a $\sqrt{n} \times \sqrt{n}$ regular grid graph the data traffic is shown to be $\Omega(n^{1+\alpha/2})$, $\alpha \leq 1$.

Partitioning schemes that are variations of block assignment scheme are described and it is shown that the data traffic generated by these schemes are asymptotically optimal. The schemes allow efficient use of up to $O(n^2)$ processors in the dense case and up to $O(n)$ processors in the sparse case before the total data traffic reaches the maximum value of $O(n^3)$ and $O(n^{3/2})$, respectively. It is shown that the block based partitioning schemes allow a better utilization of the data accessed from shared memory and thus reduce the data traffic than those based on column-wise wrap around assignment schemes.

[1]This research was supported by the National Aeronautics and Space Administration under NASA Contract Nos. NAS1-18107, NAS1-18605, and by the Air Force Office of Scientific Research under Contract No. AFOSR 88-0117 while the second author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665.

1. Introduction

Consider the problem of solving a system of linear equations

$$A\mathbf{x} = \mathbf{b}$$

where A is an $n \times n$ symmetric, positive definite coefficient matrix, $\mathbf{x}$ is an $n \times 1$ vector of variables, and $\mathbf{b}$ is an $n \times 1$ vector of constants. Applying the Cholesky decomposition to A yields

$$A = LL^T$$

where L is lower triangular with positive diagonal elements [10]. From this factorization, the solution to the system of equations is obtained by solving the triangular systems

$$L\mathbf{y} = \mathbf{b}$$

and

$$L^T\mathbf{x} = \mathbf{y}.$$

Recently, efforts have been reported for efficiently parallelizing the various steps in computing the solution of the dense and sparse systems. Most of this work has concentrated on developing algorithms that extract as much parallelism as possible on specific architectures [14,18,19,1,5,3,9]. The main emphasis there is on distributing the computational load as evenly among the processors as possible and little attention is paid towards the data traffic complexity.

In this paper we are interested in the parallel Cholesky decomposition schemes with minimum data traffic for factoring dense and sparse symmetric, positive definite matrices. The model of computation assumed for this purpose is that of a multiprocessor system with two level memory hierarchy such that each processor has local memory and all processors have access to a common shared memory. Accessing any nonzero element in the shared memory is assumed to generate a unit data traffic. No data traffic is generated in accessing the local memory. The total number of shared memory accesses from the beginning to the end of the algorithm is defined as the ***communication requirement*** or ***total data traffic*** of that algorithm implemented on the multiprocessor system.

In [17] the communication requirement of the Gaussian elimination algorithm implemented on three different architectures is analyzed. For a bus architecture where a data element may be broadcast to all the processors in one step and counts as one unit data traffic independent of the number of processors receiving the data, the data traffic complexity is shown to be $\Omega(n^2)$. For a nearest neighbor ring network, where each transmission of a data element across a link of the ring counts as one unit data traffic, the data traffic complexity is shown to be $\Omega(n^2 \cdot p)$, where p is the number of processors on the ring. The data traffic complexity for a nearest neighbor grid network is shown to be $\Omega(n^2\sqrt{p})$. In all the cases it is assumed that no element is computed in more than one processor; i.e., recomputation is not permitted. By using a different proof technique than that given in [17], it is shown here for the assumed model of computation that the data traffic complexity of the Cholesky factorization scheme is $\Omega(n^2\sqrt{p})$. The proof for the lower bound holds even if recomputation is allowed provided each processor is assigned at least $n^3/6p$ amount of work. Although we do not prove it, our result holds for other variations of Gaussian elimination as well.

In [4], a parallel sparse factorization scheme is given for local memory multiprocessor systems. This scheme has a total data traffic of $O(n^{1+\alpha}\log_2 n)$ using n^α processors. This result is improved to $O(n^{1+\alpha})$ in [7]. In this paper we present a factorization scheme that has a total data traffic of $O(n^{1+\alpha/2})$. Our main results and the organization of the paper is as follows.

In the following section, the data dependencies involved in the Cholesky factorization of a dense matrix are discussed and a parallel assignment scheme is presented. It is shown that the data traffic associated with that scheme is $O(n^2 \cdot \sqrt{p})$ when an $n \times n$ dense matrix is factored using p processors. By giving a proof on the lower bound for the data traffic, in Section 2.4 it is shown that under the condition of uniform load distribution the computation time and data traffic complexities of the assignment scheme are asymptotically optimal. In Section 3., the case of factoring sparse, symmetric, positive definite matrices is considered. The sparse matrices considered here are restricted to only those matrices that represent the graphs arising in finite difference and finite element applications. In Section 3.5, a block based parallel factoring scheme for sparse matrices is presented. The data traffic in factoring an $n \times n$ sparse matrix corresponding to a 2-dimensional regular grid graph is shown to be $O(n \cdot \sqrt{p})$. In Section 3.6 a lower bound on the data traffic in factoring the sparse matrix is shown to be $\Omega(n \cdot \sqrt{p})$. These results can be extended to other graphs that satisfy an $f(n)$-separator theorem [13]. Preliminary versions of the results given here appear in [15] and [16].

For the sake of clarity, in the following discussion the dense matrix is assumed to be of size $m \times m$ and the sparse matrix of size $n \times n$.

2. Parallel factoring of dense symmetric, positive definite matrices

The basic algebraic scheme considered here for factoring an $m \times m$ symmetric, positive definite matrix A is the *column version* of the Cholesky decomposition method [10]. An outline of this algorithm is given next and the data dependencies are discussed. Following that a partitioning scheme with optimal data traffic is presented.

In the following discussion, values in row i refer to the values of the elements on and to the left of the diagonal. Similarly, $a_{i,*}$ $(a_{*,j})$ represents all the elements in row i (in column j)

of the lower triangular part of the matrix under consideration.

2.1 The Cholesky factorization

Let $A = LL^T$; $a_{i,j} \in A$ and $l_{i,j} \in L$.

```
for $j = 1$ until $m$ do
    begin
        Initialize $l_{i,j} = a_{i,j}, \quad i = j, \cdots, m$
        for $k = 1$ until $j - 1$ do
            for $i = j$ until $m$ do
                $l_{i,j} = l_{i,j} - l_{i,k} * l_{j,k}$ ;
        $l_{j,j} = \sqrt{l_{j,j}}$ ;
        for $k = j + 1$ until $m$ do
            $l_{k,j} = l_{k,j} / l_{j,j}$ ;
    end
```

In the above algorithm for clarity, the values of $l_{i,j}$ are shown separately from those of $a_{i,j}$. In practice $l_{i,j}$ may overwrite $a_{i,j}$.

Clearly, in the Cholesky factorization scheme outlined above, computing the elements in a column j of L requires values of the elements in columns 1 through $j - 1$ of L and the values of the off-diagonal elements in j are used for computations of elements in columns $j + 1$ through m of L. Specifically, computing an element $l_{i,j}$ in L requires all the values from the set,

$$\Delta_{i,j} = \{l_{j,j'} \mid 1 \leq j' \leq j\} \cup \{l_{i,j'} \mid 1 \leq j' < j\} \cup \{a_{i,j}\}.$$

Moreover, the steps of the innermost loop, where a product of two elements of L is subtracted from $a_{i,j}$, may be performed in any order. Once $l_{i,j}$ is computed, it is used in the computation of every element in the set,

$$\bar{\Delta}_{i,j} = \{l_{i,j'} \mid j < j' \leq i\} \cup \{l_{j',i} \mid i \leq j' \leq m\}.$$

Again the element $l_{i,j}$ may be used in any order in the computations of the elements in the set $\bar{\Delta}_{i,j}$.

2.2 A partitioning scheme for Cholesky factorization

Without loss of generality, assume that $p = (r^2 + r)/2$ where r is an integer. The lower triangular part of the matrix A is divided into p partitions by taking r vertical and r horizontal sections each of size s, where $s = m/r$. All except r of the resulting p partitions are square blocks of size $s \times s$. The remaining r partitions which lie on the diagonal of the matrix are $s \times s$ triangular blocks. Each of the partitions is assigned to a single processor. Initially, each processor reads the data for its partition from shared memory into its local memory. The computations proceed in parallel according to the column version Cholesky algorithm as follows. The r processors in charge of the partitions containing the left most $s \times s$ blocks of the matrix commence the computations of their part of the factorization. As soon as an element of the factor is computed, it is written into shared memory for access by other processors. As the necessary data becomes available, the remaining processors initiate computations on the blocks assigned to them. This is continued until the entire factor is computed and written into the shared memory. This partitioning and factorization scheme for dense symmetric, positive definite matrices is referred to as the *block oriented column Cholesky*-factorization scheme or simply as the BLOCC scheme.

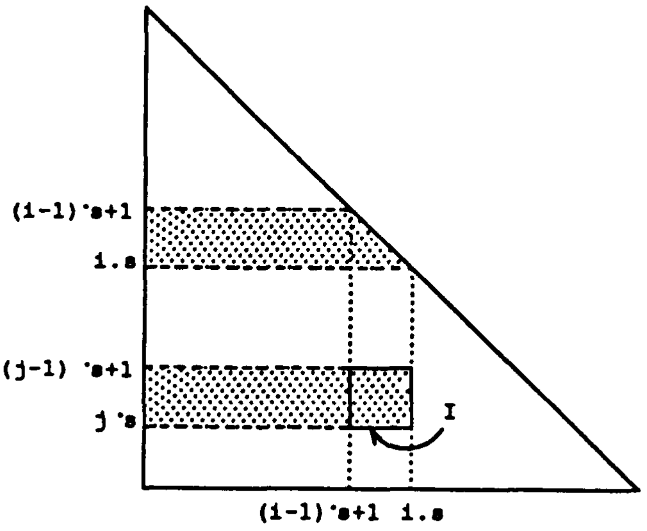


Figure 1: The data traffic associated with block I

2.3 Data traffic complexity of the BLOCC scheme

First consider the data traffic associated with computing the elements in a generic square block I in the factor L shown in Figure 1. In that figure, the darkened area represents the data elements that are required for the computations in block I. Let block I be bounded by the columns $(i-1)s+1$ and $i \cdot s$, and by the rows $(j-1)s+1$ and $j \cdot s$ where $1 \leq i \leq r$ and $1 \leq j \leq r$. The following lemma provides bounds for the communication cost of block I.

Lemma 1 *A total data traffic of $(2i-1/2)\cdot s^2+s/2$ is necessary and sufficient for computing the elements in the square block I; it is the same for all square blocks bounded by the columns $(i-1)s+1$ and $i \cdot s$. The data traffic associated with a $s \times s$ triangular diagonal block bounded by the columns $(i-1)s+1$ and $i \cdot s$, is $(i-1/2)\cdot s^2+s/2$.*

Proof: See [15]. ∎

Using these results, a bound on the total data traffic is obtained, as shown next.

Theorem 1 *The total data traffic associated with the BLOCC scheme for factoring an $m \times m$ dense symmetric, positive definite matrix using p processors is $O(m^2\sqrt{p})$.*

Proof: The total data traffic associated with all the blocks bounded by columns $(i-1)s+1$ and $i \cdot s$, $1 \leq i \leq r$, is given by,

$(r-i) \times$ (data traffic associated with a square block)

$+$ (data traffic associated with a triangular block bounded by the given columns).

From Lemma 1 and the fact that there are r such column partitions, we get the total data traffic involved in factoring the $m \times m$ dense matrix using the BLOCC scheme as:

$$\sum_{i=1}^{r}\Big((r-i)((2i-1/2)\cdot s^2+s/2)+((i-1/2)\cdot s^2+s/2)\Big).$$

Ignoring the lower order terms, the total data traffic is

$$\begin{aligned} &= \sum_{i=1}^{r}(2r\cdot i-2i^2)\cdot s^2 \\ &\leq r^3\cdot s^2/3. \end{aligned}$$

Since $p=(r^2+r)/2$ and $s=m/r$, the total data traffic is $O(m^2\sqrt{p})$. ∎

2.4 A lower bound on the data traffic complexity

Theorem 1 gives an upper bound on the data traffic associated with factoring the $m \times m$ matrix using the BLOCC assignment scheme. In the next theorem a lower bound on the data traffic in factoring the dense symmetric, positive definite matrix is established. Before giving the proof, we again consider the data dependencies involved in computing an element of the factor. Consider the computations at any element $a_{i,j}$ as shown in Figure 2. To compute the corresponding element $l_{i,j}$, values at all the elements in row j and the values of elements in column 1 through j of row i are needed. Thus, if $a_{i,j}$ is an off diagonal element then $2j$ values are needed for the computations and if it is a diagonal element (i.e., $i = j$) then j values are required. There are three observations to make regarding these computations as follows:

i. The values at all the elements in any row i are needed to complete the computations corresponding to the diagonal element $a_{i,i}$; no other values are needed for computing $l_{i,i}$. Moreover, no other element in the factor can be computed by knowing only the values in row i.

ii. If i and j are any two rows such that $i > j$, then the values of the elements in these two rows are used to complete computations at exactly one off-diagonal element $a_{i,j}$. Values from no other row are needed to complete the computations at that element.

iii. For the computations at a subset of elements spread over k_r rows and k_c columns, values from at least $\max(k_r, k_c)$ rows are needed.

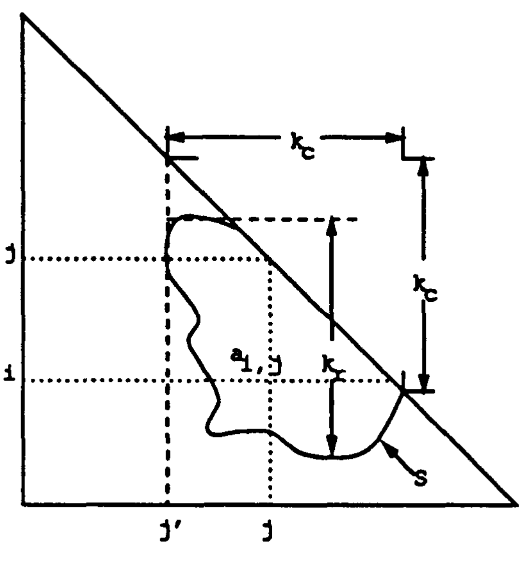


Figure 2: Data dependencies for the computations at $a_{i,j}$ and at elements in subset S

The last observation follows from the fact that the computations require values from k_r rows as well as from the k_c rows that correspond to the k_c columns. However, some of the

k_r rows and the k_c rows may overlap. In addition, note that if j' is the leftmost of the k_c columns then at least all the values in columns 1 through j' on $\max(k_r, k_c)$ rows are required in the computation of the elements in the subset under consideration (see Figure 2).

The above observations and the result established in the following lemma are used to get a lower bound on the data traffic in computing the Cholesky factor.

Lemma 2 *Let W be the amount of computational work which is to be distributed uniformly among p processors and let α be any constant less than one. For any subset of this computation consisting of $W/2$ amount of work, there are at least $(1-\alpha)\cdot p/(2-\alpha)$ processors each assigned $\alpha \cdot W/2p$ or more work from that subset.*

Proof: The work is uniformly distributed among p processors and hence each processor is assigned W/p amount of work. Now let S be a subset consisting of $W/2$ amount of computational work. All p processors may be assigned some portion of work from S. Let w_i be the computational work from S assigned to processor p_i, where $0 \leq w_i \leq W/p$. Therefore,

$$\sum_{i=1}^{p} w_i = \frac{W}{2}$$

and $W/2p$ is the average work from S performed by each processor. Thus, there is at least one processor that is assigned $W/2p$ or more work from S. Let α be a constant less than one and suppose that x processors are assigned at least $\alpha \cdot W/2p$ amount of work from S. Each of the x processors may be assigned at most W/p work from S. Now there are $p-x$ processors that compute less than $\alpha \cdot W/2p$ amount of work from S. Therefore,

$$\frac{W}{p} \cdot x + \frac{\alpha \cdot W}{2p} \cdot (p-x) \geq \frac{W}{2}.$$

Solving the inequality for x we get,

$$x \geq \frac{1-\alpha}{2-\alpha} \cdot p.$$

Thus, there are at least $(1-\alpha)\cdot p/(2-\alpha)$ processors each computing $\alpha \cdot W/2p$ or more amount of work from S. ∎

In the following theorem a bound on the data traffic associated with computing the Cholesky factor of an $m \times m$ matrix is established. Note that the result holds under stronger conditions than required by the model of computation assumed here.

Theorem 2 *Let A be a dense, $m \times m$ symmetric positive definite matrix that resides in the common memory. If the computational work is uniformly distributed among p processors, then the data traffic involved in computing the Cholesky factor of A is $\Omega(m^2 \cdot \sqrt{p})$. For $p \geq 6$, the data traffic is $\Omega(m^2 \cdot \sqrt{p})$ even if the initial values of matrix A are in the processor local memory before the computation begins.*

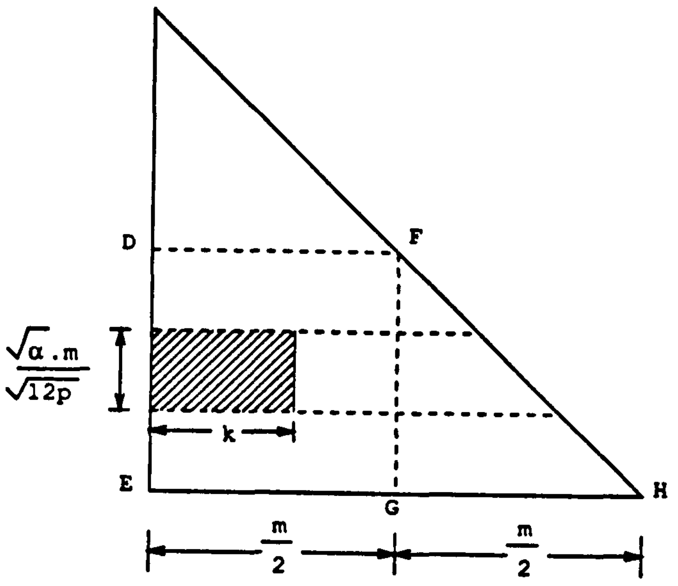


Figure 3: Data traffic associated with factoring of elements in region *FGH*

Proof: Suppose that matrix A is initially stored in the common memory. The initial value of each element is fetched at least once by the processors for computing the factor. Thus, at least $m^2/2$ amount of data traffic is associated with the Cholesky factorization. Hence when the number of processors is a constant, there is nothing to prove.

In the following it is proved that even if the matrix A is distributed initially among the processors according to their work assignment, the data traffic is $\Omega(m^2 \cdot \sqrt{p})$ when the number of processors is greater than $16(\alpha + 4/\alpha - 4)/3$ for any constant α less than one. Since there exists an α less than one such that $16(\alpha + 4/\alpha - 4)/3$ is less than six, the proven result holds for all $p \geq 6$. Consider the computations corresponding to the elements in the set $S = \{a_{i,j} \mid i, j > m/2\}$. In Figure 3 the region FGH denotes this set of elements. The total computational work in factoring the $m \times m$ matrix is $m^3/6 + m^2/2 + m/3$ and that corresponding to the elements in set S is $m^3/12 + m^2/4 + m/6$. Thus, the amount of work associated with S is exactly half of the total work. If α is a constant less than one, then from Lemma 2, there are at least $(1 - \alpha) \cdot p/(2 - \alpha)$ processors each computing at least $\alpha \cdot m^3/12p$ amount of work from the region FGH.

Let Π be the set of processors each with at least $\alpha \cdot m^3/12p$ amount of work from the region FGH and let $p_i \in \Pi$. Now the computational work associated with any element in FGH is at most m (work corresponding to element $a_{m,m}$). Therefore at least $\alpha \cdot m^2/12p$ elements in the region FGH are assigned to processor p_i. Let x be the number of rows on which the elements assigned to processor p_i lie. This implies that there are at least $\lceil(\alpha \cdot m^2)/(12 \cdot p \cdot x)\rceil$ columns on which the elements assigned to p_i lie. Therefore, from the

Observation (iii) above, data from at least $\max(x, \lceil(\alpha \cdot m^2)/(12 \cdot p \cdot x)\rceil)$ number of rows in the region $DFGE$ are required for completing the computations from region FGH assigned to processor p_i. Without loss of generality, assume that the quantity $12 \cdot p \cdot x$ divides $\alpha \cdot m^2$ evenly. Now, the quantity $\max(x, (\alpha \cdot m^2)/(12 \cdot p \cdot x))$ is minimum when $x = \sqrt{\alpha} \cdot m/\sqrt{12p}$. Thus, the computations in processor p_i require at least all the values of the elements on the $\sqrt{\alpha} \cdot m/\sqrt{12p}$ rows in region $DFGE$. In region $DFGE$ each row has $m/2$ elements and thus the values of at least $\sqrt{\alpha} \cdot m^2/(4 \cdot \sqrt{3p})$ elements from the region $DFGE$ are needed in processor p_i for performing computations in the region FGH.

Now processor p_i may also be assigned some work from the region $DFGE$ in addition to that in FGH. Hence to complete the proof, it is necessary to show that of the $\sqrt{\alpha} \cdot m^2/(4 \cdot \sqrt{3p})$ elements needed by processor p_i at least $c \cdot m^2/\sqrt{p}$ elements are not available locally, where c is a constant less than one. In that case the data traffic associated with processor p_i is at least $c \cdot m^2/\sqrt{p}$ and since there are at least $(1-\alpha) \cdot p/(2-\alpha)$ such processors, the total data traffic in computing the Cholesky factor of an $m \times m$ dense matrix is $\Omega(m^2 \cdot \sqrt{p})$. We complete the proof by showing in the following that p_i accesses at least $c \cdot m^2/\sqrt{p}$ non-local elements from region $DFGE$ for completing the computations in the region FGH.

Processor p_i is assigned at least $\alpha \cdot m^3/12p$ amount of work from the region FGH. Since each processor is assigned $m^3/6p$ amount of work (the uniform load distribution condition), p_i performs at most $(2-\alpha) \cdot m^3/12p$ amount of work in the region $DFGE$. The data traffic associated with processor p_i in completing the work in the region FGH is a minimum when all the elements from region $DFGE$ assigned to p_i lie on the $\sqrt{\alpha} \cdot m/\sqrt{12p}$ rows. Furthermore, to reduce the data traffic, as many elements on these rows as possible should be assigned to processor p_i. Now the computational work corresponding to any element $a_{i,j}$ is j; that is the work associated with an element on the leftmost column of the matrix is the smallest and it increases for elements on any row from left to right. Therefore the data traffic associated with p_i is a minimum when it is also assigned the computational work corresponding to the elements in the leftmost columns on the chosen rows of region $DFGE$. Let k be the number of the leftmost columns on which the the elements from region $DFGE$ that are assigned to p_i lie. The shaded region shown in Figure 3 corresponds to the elements which minimize the data traffic for processor p_i. Since processor p_i performs at most $(2-\alpha) \cdot m^3/12p$ amount of work in $DFGE$, the condition on k is given by,

$$\frac{\sqrt{\alpha} \cdot m}{\sqrt{12p}} \cdot \sum_{i=1}^{k} i \leq \frac{(2-\alpha) \cdot m^3}{12p}$$

i.e.,

$$k \leq \frac{1}{2}\sqrt{1 + \frac{4(2-\alpha)}{\sqrt{\alpha}} \cdot \frac{m^2}{\sqrt{3p}}} - \frac{1}{2}.$$

It can be verified that there is a constant β greater than one, such that if p is greater than $16(\alpha - 4 + 4/\alpha)/3$, then the right hand side of the above inequality is at most $m/2\beta$ for all values of m. This gives a bound on k. Therefore work corresponding to at most $(\sqrt{\alpha} \cdot m^2)/(4\beta \cdot \sqrt{3p})$ elements in the region $DFGE$ may be assigned to processor p_i which will minimize its data traffic for the computation in the region FGH. Hence of the $(\sqrt{\alpha} \cdot m^2)/(2 \cdot \sqrt{12p})$ elements needed by processor p_i for completing the computation in the region FGH, at least $(1 - 1/\beta) \cdot \sqrt{\alpha} \cdot m^2/(4\sqrt{3p})$ elements are not available locally. Thus, if the number of processors, p, is greater than $16(\alpha - 4 + 4/\alpha)/3$ for any α less than one, then the data traffic associated with processor p_i is at least $c \cdot m^2/\sqrt{p}$ for some constant c less than one. Since there are at least $(1 - \alpha) \cdot p/(2 - \alpha)$ such processors, the result follows. ∎

2.5 Remarks on the BLOCC Scheme

Assuming that each step of the innermost loop in the Cholesky decomposition costs one computational time unit and ignoring the costs associated with other steps, the sequential computation time for factoring the $m \times m$ matrix A is $m^3/6 + O(m^2)$. The BLOCC scheme described above has a computation time of $m^3/2p + O(m^2/p)$, where p is the number of processors used. As shown in Theorem 1 the associated data traffic is less than $\sqrt{2} \cdot m^2 \cdot \sqrt{p}/3$. Thus, the time and the data traffic complexities of the BLOCC scheme are optimum in an order of magnitude sense. However, the computational load in the BLOCC assignment scheme is not perfectly balanced. The processors that compute elements in the partitions that are towards the left side of the matrix L finish computation earlier than those that are on the right. This balance may be improved in several different ways, but at the cost of increasing the data traffic. In one such scheme the columns of the matrix are assigned to each processor in a wrap around fashion; that is, columns $i, p + i, \ldots, m - p + i$ are assigned to processor i. All the elements on any column of L are computed by a single processor. Let this assignment scheme be referred to as the ***wrap around assignment scheme.*** In this scheme the computation is distributed more evenly among the processors than that in the BLOCC scheme. The computation time is reduced to $m^3/6p + O(m^2)$, provided m is at least $p(p + 3)/2$. However the data traffic associated with this scheme is $m^2 \cdot p/2$, which is suboptimal. In [11] and [2] the wrap around assignment scheme is recommended as a preferred method for computing the factor on a multiprocessor system because of its good load balancing properties. Their analysis does not take into account the cost of the associated data traffic, which must be taken into account for reducing the overall execution time.

3. Parallel factoring of sparse, symmetric positive definite matrices

In this section a partitioning and assignment scheme is presented that computes the factors of an $n \times n$ matrix, associated with a 2-d regular grid graph, using n^α, $\alpha \leq 1$, processors with a total data traffic of $O(n^{1+\alpha/2})$. Then it is shown that the data traffic in factoring the matrix is $\Omega(n^{1+\alpha/2})$ when the load is distributed uniformly among n^α processors, $\alpha \leq 1$.

It is also shown that in any scheme that requires n^α processors, $\alpha \geq 1$, the data traffic is $O(n^{3/2})$.

For factoring a sparse matrix efficiently proper ordering of the matrix is essential. Ordering of the matrix to be factored also determines the data dependencies and hence the data traffic associated with any partitioning and assignment scheme. For matrices associated with regular grid graphs, nested dissection is a well known ordering scheme [6]. In the following a few basics of this ordering scheme are briefly described and some notation is introduced that is necessary for the analysis presented later. In the following it is assumed that the reader is familiar with the elementary concepts underlying the nested dissection algorithms, and the terms such as elimination order and the fill associated with the elimination process. It is also assumed that the reader is familiar with the basic graph theory concepts related to matrix representations of systems of equations, in particular, the notion of vertices, edges, separators, subgraphs of a graph, and the correspondence between the vertices and the rows and columns of the matrix, between the edges and and the non-zero elements, and the added edges and the fill-in during the factorization of the matrix. For details see [12] and [6] and the references therein.

3.1 Nested dissection method as applied to 2-d grid graphs

A nested dissection method may be viewed as a divide-and-conquer algorithm on an undirected graph. It relies on finding a small set of vertices, called the separator set, in the graph such that the removal of these vertices divides the graph approximately in half. Informally, the nested dissection method orders the vertices of the graphs as follows. The vertices in the separator set are ordered last. Then the vertices in the subgraphs obtained from the original graph by removing the separator are ordered recursively. In [12] a nested dissection algorithm is given for ordering the vertices of any graph G such that G and all subgraphs of G satisfy a $\sqrt{n}$-separator theorem. The ordering produced by this algorithm guarantees a $O(n \log n)$ fill and $O(n^{3/2})$ sequential operation count for a system corresponding to an n-vertex graph G. In [8] a nested dissection algorithm is given for ordering the vertices of a graph G that has a $\sqrt{n}$-separator decomposition.* For a detailed treatment of the nested dissection methods and for the relevant practical applications see [6].

For the sake of simplicity and clarity, here only the systems corresponding to $\sqrt{n} \times \sqrt{n}$ regular grid graphs are analyzed. However, the techniques developed for analyzing data traffic complexities are applicable to other systems where the nested dissection method can be used to give a "good" ordering. In the following, the nested dissection method used for ordering the vertices in a $\sqrt{n} \times \sqrt{n}$ regular grid graph is briefly described. In the discussion, the grid graph is sometimes simply referred to as the grid and a subgraph of the grid graphs is referred to as a subgrid. For the rest of the discussion, it is also assumed that

*A graph G is said to have a *$\sqrt{n}$-separator decomposition* for constants $\alpha < 1$ and $\beta > 0$ if G has a $\sqrt{n}$-separator C and every connected component of $G - C$ has a $\sqrt{n}$-separator decomposition.

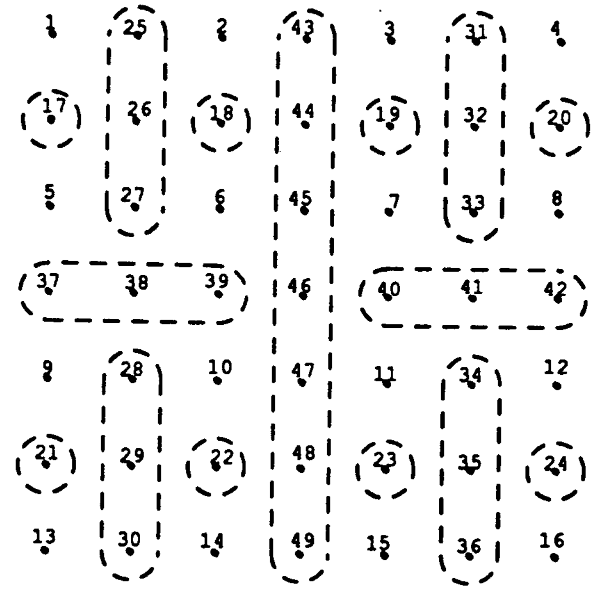


Figure 4: A 7 × 7 grid with nested dissection ordering

the vertices of the grid are connected according to a 9-point stencil, unless otherwise stated. Let V be the set of vertices of a $\sqrt{n} \times \sqrt{n}$ regular grid graph. Without loss of generality, assume that $\sqrt{n} = 2^l - 1$ for some integer l. Let S_0 be the set of $2^l - 1$ vertices on a vertical mesh line, the removal of which partitions V into two subgrids, V_1 and V_2 such that the vertices of both the subgrids are arranged in a $(2^l - 1) \times (2^{l-1} - 1)$ mesh. The vertices of S_0 are numbered from $n - \sqrt{n} + 1$ to n in any order. Suppose that V_1 is the left subgrid and V_2 is the right subgrid. Let S_1 be the set of vertices on a horizontal mesh line that divides V_1 into two equal parts each containing $(2^{l-1} - 1)^2$ vertices that are arranged along a $(2^{l-1} - 1) \times (2^{l-1} - 1)$ square mesh. Similarly, let S_2 be the set of vertices from V_2 which, when removed, produce two equal halves from V_2. Both S_1 and S_2 contain $2^{l-1} - 1$ vertices. Let the vertices in S_1 be numbered from $n - 2\sqrt{n} + 2$ to $n - 3\sqrt{n}/2 + 1/2$ and those in S_2 be numbered from $n - 3\sqrt{n}/2 + 3/2$ to $n - \sqrt{n}$. Thus, the removal from V of the vertices in the set $S_0 \bigcup S_1 \bigcup S_2$ partitions V into four $(\sqrt{n} - 1)/2 \times (\sqrt{n} - 1)/2$ subgrids. The separator set $S_0 \bigcup S_1 \bigcup S_2$ is referred to as the "+"-*separator* for the grid corresponding to V. The middle vertical part of the "+"-separator is referred to as the ***vertical sub-separator*** and each of the two horizontal halves of the "+"-separator is referred to as the ***horizontal sub-separator***. All the vertices of the four subgrids are numbered by recursively identifying and ordering the vertices on the "+"-separators of the subgrids induced by the vertices ordered so far. The recursion stops when a subgrid has only one vertex on it. For any "+"-separator, there is a vertical sub-separator and two horizontal sub-separators. With the above described ordering scheme, for any given "+"-separator, the vertices on the two horizontal sub-separators are given numbers that are smaller than those assigned to the vertices on the corresponding vertical sub-separator. Thus, we say that the vertices on a horizontal sub-separator are ***ordered ahead*** of the vertices on the corresponding vertical sub-separator or that the vertices on the vertical sub-separator are ***ordered after*** those on the horizontal sub-separators. An example of ordering the vertices in a 7 x 7 grid is shown in Figure 4. Observe that the grid is recursively partitioned into four subgrids by a set of vertices that form a "+"-separator.

To label the subgrids and the separators of the grid graph, we use the notation given in [7]. Each subgraph and the separator that induces the subgraph are given a level number depending on the recursion level of the nested dissection on which the subgraph is ordered. Under this scheme the original grid is called a *level*-0 (sub)grid. The four subgrids of size $(\sqrt{n}-1)/2 \times (\sqrt{n}-1)/2$ are the level-1 subgrids. The "+"-separator that partitions the level-0 grid into the four level-1 subgrids is called the level-1 "+"-separator or simply as the level-1 separator. Thus, if n is equal to $(2^l-1)^2$, there are l levels of subgrids numbered 0 through $l-1$ and $l-1$ levels of separators, numbered 1 through $l-1$.

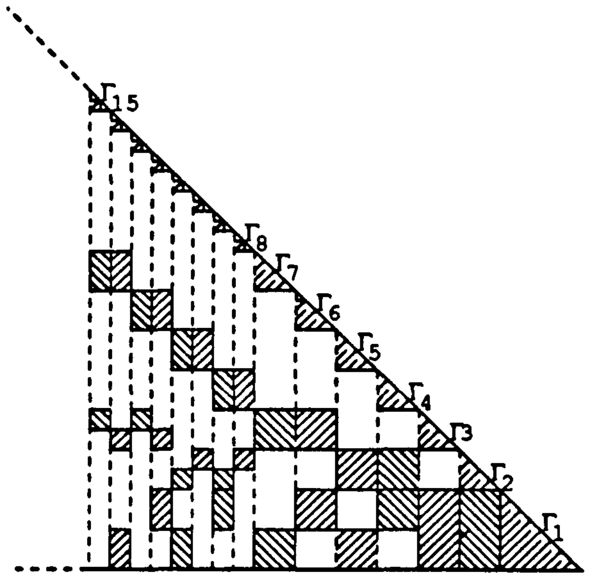
In the following it is assumed that the matrix to be factored is ordered using the nested dissection scheme and that the symbolic factorization step is already completed.

3.2 Cholesky factorization scheme revisited

Consider the Cholesky factorization scheme described in Section 2.1 for factoring a sparse symmetric, positive definite matrix.

Clearly, the main difference between factoring a sparse and a dense matrix using the Cholesky factorization scheme is that in the former case there is no need to modify column j by all columns to the left of it. Specifically, column j is modified only by columns k for which $l_{j,k} \neq 0$. Moreover, if column k modifies column j, only the nonzero elements of column k need to be fetched. Exactly which elements are needed is formalized later. In Figure 5(a), the zero-nonzero structure of L, corresponding to the vertices of the separators on the first two levels, is shown schematically. The shaded areas represent the nonzeros. The corresponding grid is shown in Figure 5(b). It is clear from the figure that only certain values from certain columns are needed for computing an element of the factor.

Another important difference is that, because of the ordering applied, several columns may be computed simultaneously. As stated earlier, column i and row j of the matrix corresponds to a vertex v_i in the elimination graph and the factoring of the matrix corresponds to the elimination of the vertices. Thus, all the vertices on the level $l-1$ subgrids may be eliminated simultaneously followed by those on the level $l-2$ and so on. This observation is useful in extracting parallelism in the factorization step.



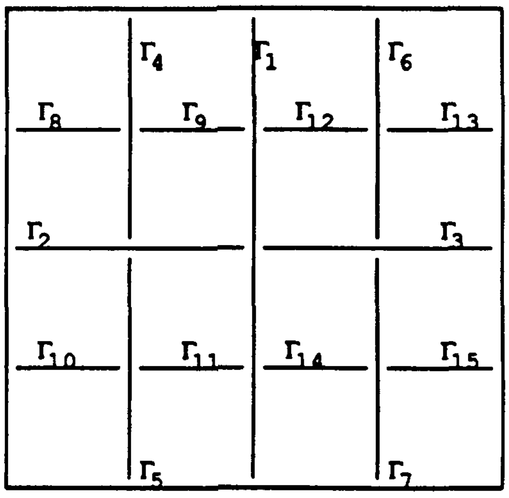


Figure 5: Structure of L

3.3 The worst case data traffic complexity

In this section a bound on the worst case data traffic complexity for factoring the matrix A is established. Clearly, the communication requirement is the worst when the use of local memory is not allowed. Thus, an upper bound on the worst case data traffic is obtained by assuming that the values of all the elements of the lower triangular part of matrix A and those of L, as well as any intermediate results, are stored in the shared memory. Suppose also that any number of processors are allowed to participate in computing a nonzero element of the factor provided that no computation is repeated. Consider the computations associated with a nonzero element $l_{i,j} \in L$. Recall that in computing $l_{i,j}$, first $a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} \cdot l_{j,k}$ is evaluated and then the resulting value is divided by $l_{j,j}$. Thus, for each multiplication, there is one subtraction operation, at most one division and three memory references and a constant overhead such as index computation. Therefore, in the worst case, each multiplication operation in the Cholesky factorization is associated with a constant amount of data traffic. The following theorem gives a bound on the worst case total data traffic. In the proof of the theorem, the result given in Theorem 8.1.8 of [6] is assumed. That theorem states that the number of operations required to factor a matrix associated with an n-vertex 2-D grid ordered by nested dissection is given by $829n^{3/2}/84 + O(n \cdot \log n)$. Although the following result is obvious, it is useful because it is independent of the number of processors used and it gives the worst case bound on the data traffic even for the models of computation that are more restrictive.

Theorem 3 *The worst case data traffic associated with factoring the matrix A is $O(n^{3/2})$.*

Proof: Associated with each multiplication operation in the factorization there are at most a constant number of memory references. Suppose that k memory references are involved per multiplication. Thus, the total data traffic is

$$\leq k \cdot \text{number of multiplication operations.}$$

Now, the number of multiplication operations associated with factoring matrix A is $O(n^{3/2})$ [6]. Hence, the worst case total data traffic is $O(n^{3/2})$. ∎

Note that the above theorem is applicable to all the graphs for which a $\sqrt{n}$-separator theorem holds.

3.4 Data dependencies for the sparse Cholesky factorization

The worst case bound on the data traffic established in Theorem 3 can be improved for the model of architecture assumed in the case of the dense matrices. In that model, no element is fetched more than once from the shared memory and hence the values of the elements used in more than one operation are stored in the local memory associated with the processor. To maximize the potential of such a model, it is necessary to clearly understand the data dependencies involved. The vertices of the grid are ordered using the recursive nested dissection scheme. Hence it is sufficient to investigate the data dependencies involved in computing the elements of L in the columns corresponding to the vertices in a generic "+"-separator. This is accomplished in the next two lemmas.

Let $\eta_i^j = \{k | k \leq j \text{ and } l_{i,k} \neq 0, l_{i,k} \in L\}$; i.e., η_i^j is the set of all columns of the factor L to the left of the column $j+1$ such that the elements in row i of these columns are nonzero. Let $\bar{\eta}_{i,k}^j = \bigcup_{s=i}^k \eta_s^j$; i.e., $\bar{\eta}_{i,k}^j$ is the set of all the columns to left of column $j+1$ such that on each of these columns there is a nonzero element in at least one of the i through k rows of the factor. Let Γ represent any m-vertex sub-separator. It is assumed that all the vertices in any sub-separator are ordered consecutively. Let $low(\Gamma)$ and $high(\Gamma)$ be the indices of the lowest and the highest ordered vertices, respectively, on the sub-separator Γ. Note that $high(\Gamma) - low(\Gamma) + 1 = m$. In Figure 6, a sub-separator Γ is shown. This sub-separator separates the vertices in regions R_1 and R_2. The diagonal and off-diagonal non-zero blocks associated with this sub-separator are shown in Figure 7.

The following lemma establishes some basic sub-separator related properties that are useful in analyzing the communication requirements.

Lemma 3 *Let Γ be any m-vertex sub-separator.* (i) *Corresponding to the vertices of Γ there is a dense $m \times m$ triangular diagonal block in the Cholesky factor.* (ii) *In the factor L, the*

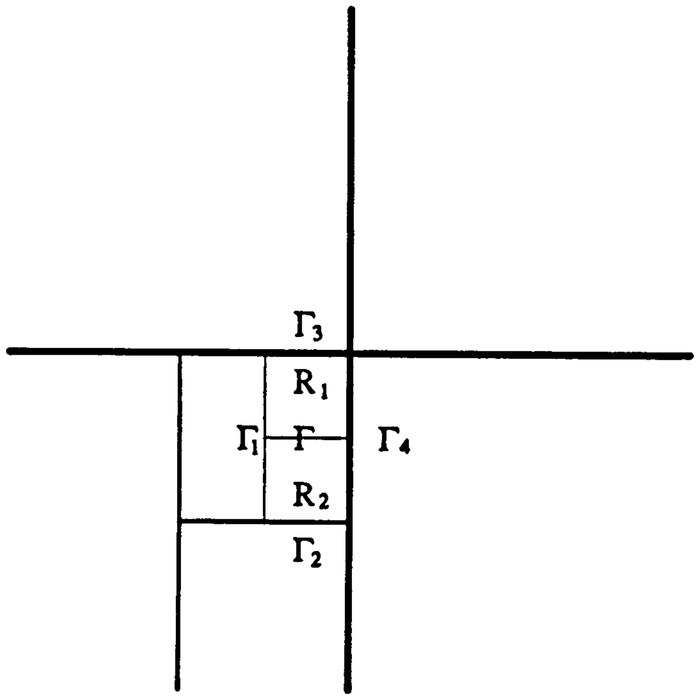


Figure 6: Sub-separator Γ with four surrounding sub-separators

columns low(Γ) through high(Γ) contain at most four off-diagonal rectangular blocks with nonzero elements. Each of these blocks is of size at most $(c_1 \cdot m + c_2) \times m$ where $c_1 \leq 2$ and $c_2 \leq 3$ are positive integer constants. Any nonzero element in these columns is either in one of these four blocks or in the diagonal triangular block.

Proof: The first part of the lemma is obvious. In Figure 6, the sub-separator Γ separates the vertices in regions R_1 and R_2. Since the vertices in these two regions are ordered ahead of those of Γ, the fill due to the elimination of vertices in regions R_1 and R_2 ensures a dense $m \times m$ triangular diagonal block bounded by columns *low*(Γ) and *high*(Γ) as shown in Figure 7.

To prove the second part of the lemma, again consider Figure 6. In that figure, the thickness of the lines qualitatively indicates the separator levels in the nested dissection ordering. Let Γ_1, Γ_2, Γ_3, and Γ_4 be the four partial sub-separators that surround the sub-separator Γ. Because of the nature of the nested dissection ordering, the vertices of Γ are "connected" to only those higher ordered vertices that lie on Γ_1, Γ_2, Γ_3, and Γ_4 and to no other vertices.[†] Thus, all the nonzeros on columns *low*(Γ) through *high*(Γ) in rows below *high*(Γ) are confined to only the rows corresponding to the vertices on Γ_1, Γ_2, Γ_3, and Γ_4. Furthermore,

[†]Vertex u is said to be "connected" to vertex v if there exists a path $[u, u_1, u_2, \ldots, u_k, v]$ of length one or more in the grid graph such that $index(u_r) < \min(index(u), index(v))$, for $1 \leq r \leq k$; in such a case, $l_{i,j} \in L$ is a non-zero, where $i = index(u), j = index(v)$.

each vertex in Γ is "connected" to every vertex on these four partial sub-separators and hence the four rectangular blocks are dense. This is shown schematically in Figure 7. It can be verified that if Γ is a horizontal m-vertex sub-separator, then the surrounding box of vertices is of dimension $(2m+3) \times (m+2)$. Therefore there are two rectangular off-diagonal dense blocks of dimension at most $(2m+3) \times m$ and the other two of dimension at most $(m+2) \times m$. Similarly, if Γ is a vertical m-vertex sub-separator, there are four off-diagonal rectangular blocks of dimension at most $(m+2) \times m$ in the factor. If Γ is not surrounded on all four sides then some of these blocks will be missing. ∎

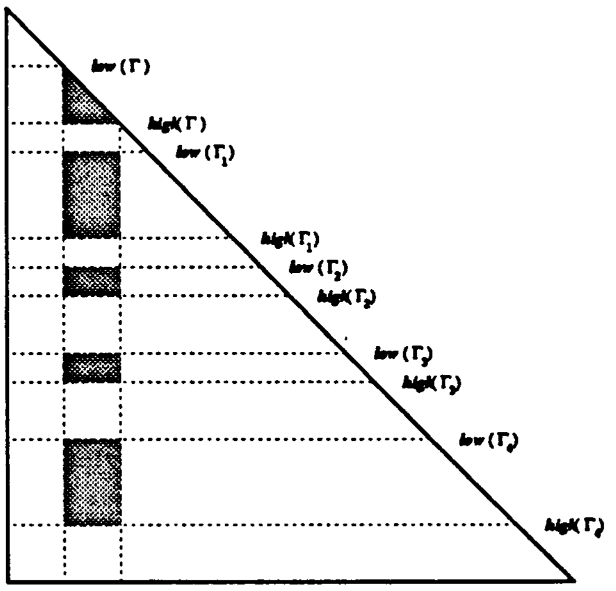


Figure 7: Off-diagonal blocks with nonzeros corresponding to sub-separator Γ

From the above lemma it is clear that, in computing the nonzero elements in the columns corresponding to the vertices on a sub-separator, only the data dependencies of the elements in the four rectangular blocks and the diagonal triangular block need be considered. This is accomplished in the following lemma where a bound is derived on the amount of data required in computing the nonzero elements lying on a given row and on one of the five blocks. The lemma shows that the number of nonzero elements in any row i of the factor L is less than $c \cdot m$ where c is an integer constant and m is the size of the sub-separator to which the vertex corresponding to row i belongs. It is then shown that, for any row i, the computations at all the elements $l_{i,j} \in L$, $low(\Gamma) \leq j \leq high(\Gamma)$, for some m-vertex sub-separator Γ, require a total of less than $c \cdot m$ nonzero elements from that row. Note that this count is independent of the sub-separator to which the vertex corresponding to row i belongs. Thus, the computations at all the elements in a row of any of the five blocks specified in Lemma 3 require only $c \cdot m$ elements from that row, irrespective of the relative location of the off-diagonal blocks in the factor.

Lemma 4 *Let Γ be any m-vertex sub-separator. The nonzero elements from row i, $i \geq low(\Gamma)$, required in completing the computations of all elements $l_{i,j} \in L$ such that $low(\Gamma) \leq j \leq high(\Gamma)$, are those elements in row i on the columns in the set given by, $\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}$. For all i greater than or equal to $low(\Gamma)$, $||\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}||$ is at most $c \cdot m$ for some constant c.*

Proof: Any nonzero element $l_{i,j} \in L$, $i \geq low(\Gamma)$ and $low(\Gamma) \leq j \leq high(\Gamma)$, is in one of the five blocks specified in Lemma 3. Hence, to prove the result of this lemma, only the rows that intersect one of these blocks need to be considered. The result for $low(\Gamma) \leq i \leq high(\Gamma)$ is proved first followed by that for $i > high(\Gamma)$.

When $low(\Gamma) \leq i \leq high(\Gamma)$, $\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)} = \eta_i^{high(\Gamma)}$, since, $\eta_i^{high(\Gamma)} \subset \bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)}$. By definition, the set $\eta_i^{high(\Gamma)}$ contains all the columns that have a nonzero element in row i. Clearly, the nonzero elements from the row i required in completing the computations at all the elements $l_{i,j} \in L$, $low(\Gamma) \leq j \leq high(\Gamma)$, are on columns in the set $\eta_i^{high(\Gamma)}$.

To measure the size of the set $\eta_i^{high(\Gamma)}$, note that it is bounded by the number of vertices ordered ahead of the vertex i and which are "connected" to vertex i. Using the recursive nature of the nested dissection ordering it can be verified that in the case of constant degree grid graphs and when $low(\Gamma) \leq i \leq high(\Gamma)$, the size of the set $\eta_i^{high(\Gamma)}$ is bounded by $c \cdot m$, where c is a constant dependent both on the degree of the graph and on whether Γ is a horizontal or vertical sub-separator. If Γ is a horizontal m-vertex sub-separator then, for a 5-point stencil, c is equal to 7 and, for a 9-point stencil, c is equal to 11. When Γ is a vertical m-vertex sub-separator, the values of c are 5 and 7, respectively. This completes the proof when $low(\Gamma) \leq i \leq high(\Gamma)$.

The case where $i > high(\Gamma)$ is considered next. As shown above, $||\eta_i^{high(\Gamma)}||$ depends on the size of the sub-separator to which the vertex i belongs and hence, when $i > high(\Gamma)$, $||\eta_i^{high(\Gamma)}||$ can be much greater than $O(m)$ where m is size of Γ. However, when the computation of only those elements in row i that lie on columns $low(\Gamma)$ through $high(\Gamma)$ are of concern, each of these computations consists of a product of a nonzero element in row i and a nonzero element in one of the rows $low(\Gamma)$ through $high(\Gamma)$ in the column $high(\Gamma)$ or in some other column to the left of it. Thus, for these computations, only the columns that have a nonzero element in row i and in row j, where $low(\Gamma) \leq j \leq high(\Gamma)$, are of interest. The set $\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)}$ consists of all columns that have a nonzero element in at least one of the rows $low(\Gamma)$ through $high(\Gamma)$. Similarly, $\eta_i^{high(\Gamma)}$ consists of all the columns that have a nonzero element in row i. Clearly, the set $\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}$ consists of all the columns which contain all the pairs of nonzero elements that must be used in completing the computations at all the elements $l_{i,j}$, $low(\Gamma) \le j \le high(\Gamma)$. Thus, the nonzero elements from row i, $i > high(\Gamma)$, required in completing computations at all the elements $l_{i,j} \in L$ such that $low(\Gamma) \le j \le high(\Gamma)$, are those elements in the row i on the columns in the set given by $\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}$.

To get a bound on the size of $\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}$, consider the m-vertex horizontal sub-separator Γ shown in Figure 6. It is surrounded by sub-separators Γ_1, Γ_2, Γ_3, and Γ_4. Suppose that $low(\Gamma_1) \le i \le high(\Gamma_1)$. The set $\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}$ consists of columns corresponding to vertices on Γ or corresponding to those vertices ordered ahead of them which are "connected" to at least one vertex in Γ and to the vertex corresponding to row i. Using the recursive ordering of the nested dissection scheme it can be shown that the number of such vertices is less than $7m$. Thus, $\|\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}\| \le 7m$, for $low(\Gamma_1) \le i \le high(\Gamma_1)$. The same bound is obtained when $low(\Gamma_4) \le i \le high(\Gamma_4)$. If $low(\Gamma_2) \le i \le high(\Gamma_2)$ or $low(\Gamma_3) \le i \le high(\Gamma_3)$ then it can be verified that, $\|\bar{\eta}^{high(\Gamma)}_{low(\Gamma),high(\Gamma)} \bigcap \eta_i^{high(\Gamma)}\| \le 3m$. If Γ is vertical sub-separator the two bounds are $5m$ and $5m/2$ respectively. ∎

3.5 A partitioning scheme with minimum data traffic

In this section a partitioning scheme for computing the factor of the sparse matrix A is described. Suppose that an $n \times n$ matrix is to be factored using n^α processors, $\alpha \le 1$. The vertices of the $\sqrt{n} \times \sqrt{n}$ grid graph corresponding to this matrix are ordered using the nested dissection method described earlier. Assuming $n = (2^l - 1)^2$, the ordering results in l levels of subgraphs and $l - 1$ levels of "+"-separators. If the original $\sqrt{n} \times \sqrt{n}$ grid is considered to be on level 0, then on level i there are 2^{2i} level-i subgraphs each of size $(2^{l-i} - 1) \times (2^{l-i} - 1)$. Without loss of generality, assume that $\alpha \cdot l$ is an integer. Thus, in the partitioning scheme described here, all the vertices on a level-αl subgraph are assigned to the same processor. In that scheme, initially each processor independently computes the elements in the factor corresponding to a $(2^{(1-\alpha)\cdot l} - 1) \times (2^{(1-\alpha)\cdot l} - 1)$ subgraph which are separated from one another by the level-αl separators. Once the elements in the columns corresponding to the vertices on the level-$(l - 1)$ through level-αl separators are computed locally, a processor p_i combines with three other processors to compute the elements on the columns of L corresponding to the vertices on the level-$(\alpha l - 1)$ "+"-separator. The two horizontal sub-separators are computed by two processors and the vertical sub-separator of that level is computed by all four processors. The next lower level "+"-separator is computed in parallel by sixteen processors from the four neighboring groups. This is continued until all the vertices are eliminated. On each level of computation each group of processors computes the elements of the factor independent of the other groups. The elimination of the vertices on the vertical sub-separator of level-1 is computed in parallel by all processors. This corresponds to factoring a $\sqrt{n} \times \sqrt{n}$ dense matrix. The computations corresponding to the level-i separator, $i < \alpha \cdot l$, are performed as follows. The computations corresponding to the vertices on level-$(\alpha l - k)$ "+"-separator, $1 \leq k < \alpha \cdot l$, are completed by $p = 2^{2 \cdot k}$ processors working in parallel. Using all the available processors, the factorization corresponding to the $m \times m$ triangular diagonal block is first completed. Then the processors are used to compute the elements corresponding to the four off-diagonal blocks. For the first part, the BLOCC factorization scheme described for the dense matrices is used. The $m \times m$ dense diagonal block is partitioned into $r^2/2 - r/2$ square blocks and r diagonal triangular blocks each of size $m/r \times m/r$ where $p = r^2/2 + r/2$, and each of these p partitions is assigned to a unique processor. Each processor completes the computations corresponding to its partition by accessing the required data from the shared memory. For the purpose of factoring, the off-diagonal blocks are treated as if they were adjacent, and the resultant rectangular block is partitioned into p sub-blocks each of size $c \cdot m/\sqrt{p} \times m/\sqrt{p}$, where $c \leq 6$ for a horizontal sub-separator and $c \leq 4$ for a vertical sub-separator. Again each partition is assigned to a separate processor. This process is repeated on the next lower level "+"-separator. Thus, in the assignment scheme described here, each processor is assigned a new subblock on each level and the size of the subblock assigned to a processor varies from one level to the next. Let this partitioning scheme be referred to as the ***sparse block oriented column Cholesky*** factorization scheme or simply as the ***sparse BLOCC*** scheme. Note that the underlying numeric algorithm is the column oriented Cholesky factorization.

Data traffic associated with an m-vertex sub-separator

The sparse BLOCC scheme, described above, may be considered as a sequence of steps, each step corresponding to the elimination of vertices on the "+"-separators of some level. Initially, a single processor computes all the non-zero elements corresponding to a "+"-separator in the factor. As the computation proceeds, more than one processor work together to compute the elements corresponding to a "+"-separator. On any such step, first the non-zero elements in the columns corresponding to the horizontal sub-separators are computed and then those in the columns corresponding to the vertical part are computed. Here we analyze the data traffic associated with any one step, on which p processors combine together to compute the elements corresponding to a sub-separator.

By Lemma 3, for any sub-separator Γ there are at most five non-zero blocks in the columns corresponding to the vertices on Γ. The number of non-zero blocks is five when Γ is enclosed within a rectangular box formed by the sub-separators with vertices that are ordered after those on Γ (see Figure 6). The following lemma gives a bound on the data traffic associated with computing the elements in the columns corresponding to such sub-separators. Not all sub-separators are enclosed by such rectangular boxes. In such cases there are less elements to be computed and consequently there is less data traffic. For the sake of simplicity of the analysis, it is assumed that no element of the factor needed in the computation of the five non-zero blocks is initially in the local memory of any of the p processors. Thus, the data traffic given below is a conservative estimate.

Lemma 5 *Let Γ be any m-vertex sub-separator and p be the number of processors available for computing the elements of the factor in all the non-zero blocks within the columns $low(\Gamma)$ through $high(\Gamma)$. If Γ is an m-vertex horizontal sub-separator, then the associated data traffic is at most $(53 + 11\sqrt{2})m^2 \cdot \sqrt{p}$. If it is a vertical sub-separator, then the data traffic is at most $(28 + 8\sqrt{2})m^2 \cdot \sqrt{p}$.*

Proof: Let Γ be an m-vertex horizontal sub-separator that is enclosed completely within a rectangular box formed by the sub-separators whose vertices are eliminated after the vertices of Γ. Such a sub-separator has the worst case communication requirements among all the m-vertex sub-separators.

First, consider the data traffic associated with computing the elements of the factor in the triangular diagonal block using $p = r^2/2 + r/2$ processors. Each of the sub-blocks requires nonzero elements from at most $2m/r$ rows out of the m rows in the range $low(\Gamma)$ through $high(\Gamma)$ of the factor. No other information is needed. From the proof of Lemma 4, each of these rows has at most $11m$ nonzeros. Thus the communication requirement of each partition is at most $11m \cdot 2m/\sqrt{2p}$ and the total communication requirement of the triangular block is bounded above by $11\sqrt{2}m^2 \cdot \sqrt{p}$.

Now consider the data traffic associated with the off-diagonal blocks. Each partition is of size $6m/\sqrt{p} \times m/\sqrt{p}$. Thus, each partition requires nonzero elements from $6m/\sqrt{p}$ rows which are below the row $high(\Gamma)$ in the factor. From the proof of Lemma 4, each of these rows has at most $7m$ nonzeros that are useful in completing the computations in any of the partitions. Each partition also requires information from $m/\sqrt{p}$ rows from the region $low(\Gamma)$ through $high(\Gamma)$. Each of these rows has at most $11m$ nonzeros. Thus, the communication requirement of each partition is at most $7m \cdot 6m/\sqrt{p} + 11m \cdot m/\sqrt{p} = 53m^2/\sqrt{p}$ and the total communication requirement of completing the computations at the off-diagonal blocks using p processors is less than or equal to $53m^2\sqrt{p}$.

Adding the communication costs corresponding to the diagonal and the off-diagonal blocks we get the total data traffic associated with Γ to be less than or equal to $(53 + 11\sqrt{2})m^2 \cdot \sqrt{p}$.

A similar analysis can be used to compute the data traffic when Γ is an m-vertex vertical sub-separator and can be shown to be bounded above by $(28 + 8\sqrt{2})m^2 \cdot \sqrt{p}$. ∎

The total data traffic of the sparse BLOCC scheme

Applying the results from the above lemma, a bound is obtained on the total data traffic of the sparse BLOCC scheme. First some notation is introduced. Let $\tau_h(m, p, k)$ represent the data traffic using p processors in completing the computations at all the nonzero elements $l_{i,j} \in L$ in the columns corresponding to an m-vertex horizontal sub-separator that is surrounded by higher ordered vertices on k sides. Let $\tau_v(m, p, k)$ represent the same for an m-vertex vertical sub-separator. From Lemma 5, $\tau_h(m, p, 4)$ is at most $(53 + 11\sqrt{2})m^2 \cdot \sqrt{p}$ and $\tau_v(m, p, 4)$ is at most $(28 + 8\sqrt{2})m^2 \cdot \sqrt{p}$. Let $\tau_g(m', p, k)$ represent the total data traffic, using p processors, in completing the computations corresponding to all the sub-separators within an m'-vertex sub-grid that is surrounded by higher ordered vertices on k sides. Note that the quantities τ_h and τ_v represent the data traffic corresponding to the vertices on a horizontal and a vertical sub-separator, respectively, whereas τ_g represents the data traffic corresponding to the vertices on an entire sub-grid.

The following theorem gives an upper bound on the total data traffic in factoring the matrix A associated with an n-vertex 2-D regular grid graph using n^α processors with the scheduling scheme as described above.

Theorem 4 *The total data traffic in factoring the $n \times n$ sparse matrix A, using n^α processors, is $O(n^{1+\alpha/2})$; i.e., $\tau_g(n, n^\alpha, 0) = O(n^{1+\alpha/2})$.*

Proof: On an $n^{1/2} \times n^{1/2}$ regular grid there is an $n^{1/2}$-vertex vertical sub-separator and two $n^{1/2}/2$-vertex horizontal sub-separators (ignoring the additive constant -1). The vertical sub-separator is not surrounded by any vertices that are ordered after the vertices on the vertical sub-separator. Each of the two horizontal sub-separators are surrounded by such vertices only on one side. These three sub-separators subdivide the n-vertex grid graph into four sub-grids of size $n^{1/2}/2 \times n^{1/2}/2$, each surrounded on two sides by higher ordered vertices. Thus, the total data traffic in factoring the corresponding matrix A is given by,

$$\tau_g(n, n^\alpha, 0) = \tau_v(n^{1/2}, n^\alpha, 0) + 2\tau_h(\frac{1}{2}n^{1/2}, \frac{1}{2}n^\alpha, 1) + 4\tau_g(\frac{1}{4}n, \frac{1}{4}n^\alpha, 2).$$

A recursive expansion of the above expression contains data traffic terms for vertical sub-separators of different sizes that are surrounded on zero sides, two sides, three sides (in two different ways), and on all four sides by higher ordered vertices. It also contains data traffic expressions for horizontal sub-separators of different sizes surrounded in five different ways. To keep the analysis simple, it is assumed that all the four sub-grids of size $n^{1/2}/2 \times n^{1/2}/2$ are surrounded on all four sides. This simplification results in a conservative expression for the data traffic, but affects only the constant terms in the bound. Thus,

$$\tau_g(n, n^\alpha, 0) \leq \tau_v(n^{1/2}, n^\alpha, 0) + 2\tau_h(\frac{1}{2}n^{1/2}, \frac{1}{2}n^\alpha, 1) + 4\tau_g(\frac{1}{4}n, \frac{1}{4}n^\alpha, 4).$$

Now,

$$\tau_g(\frac{1}{4}n, \frac{1}{4}n^\alpha, 4) = \tau_v(\frac{1}{2}n^{1/2}, \frac{1}{4}n^\alpha, 4) + 2\tau_h(\frac{1}{4}n^{1/2}, \frac{1}{8}n^\alpha, 4) + 4\tau_g(\frac{1}{16}n, \frac{1}{16}n^\alpha, 4).$$

From Lemma 5, it follows that,

$$\tau_g(\frac{1}{4}n, \frac{1}{4}n^\alpha, 4) \leq \frac{134 + 85\sqrt{2}}{16} \cdot n^{1+\alpha/2}.$$

An analysis similar to that given in Lemma 5 yields

$$\tau_v(n^{1/2}, n^{\alpha}, 0) \leq 8\sqrt{2} \cdot n^{1+\alpha/2}$$

and

$$\tau_h(\frac{1}{2}n^{1/2}, \frac{1}{2}n^{\alpha}, 1) \leq \frac{22+25\sqrt{2}}{8} \cdot n^{1+\alpha/2}.$$

Thus, we get

$$\tau_g(n, n^{\alpha}, 0) \leq \frac{78+71\sqrt{2}}{2} \cdot n^{1+\alpha/2}.$$

∎

3.6 A lower bound on the data traffic complexity

In the following theorem, the communication bound of the sparse BLOCC scheme is shown, by giving a lower bound proof, to be optimal in an order of magnitude sense.

Theorem 5 *Under the condition of uniform load distribution, the data traffic in factoring the $n \times n$ sparse matrix A, using n^{α} processors, $\alpha \leq 1$, is $\Omega(n^{1+\alpha/2})$.*

Proof: For a regular 2-D grid graph with n vertices, the separator size for nested dissection ordering is $n^{1/2}$ [12]. From Lemma 3, it follows that the factor L has an $n^{1/2} \times n^{1/2}$ dense triangular diagonal block incorporated in it. From Theorem 2, the data traffic involved in completing the computations associated with the elements of this dense triangular block, under the condition of uniform load distribution using n^{α} processors, is $\Omega(n^{1+\alpha/2})$. Since the factorization of A cannot be completed without completing the factorization of this dense block, the result follows. ∎

From Theorem 4 and Theorem 5, it is clear that the load assignment scheme described here for factoring the $n \times n$ sparse matrix using n^{α} processors is optimal in an order of magnitude sense. Note that when n^{α}, $\alpha > 1$, processors are used, the data traffic bound given in Theorem 3 holds.

4. Concluding remarks

In this paper we have analyzed the data dependencies in the Cholesky factorization of dense and sparse symmetric, positive definite matrices. The model of computation assumes a multiprocessor system with a memory hierarchy. Based on this analysis it is shown that under the condition of uniform load distribution the data traffic associated with factoring an $n \times n$ dense matrix is $\Omega(n^{2+\alpha/2})$ when n^α, $\alpha \leq 2$, processors are used. The same is shown to be $\Omega(n^{1+\alpha/2})$, $\alpha \leq 1$, for factoring an $n \times n$ sparse matrix representing a 2-dimensional regular grid graph where the vertices are ordered using the nested dissection ordering methods. Block based partitioning schemes are presented that asymptotically achieve these bounds on the data traffic.

The sequential computation time for factoring the $n \times n$ sparse matrix A is $829n^{3/2}/84 + O(n \log n)$ [6]. As stated for the dense matrix case, the assumption here is that the computation cost of each step of the innermost loop is one and costs involved in the other steps are ignored. Under the same assumption, it can be shown that the computation time for the sparse BLOCC scheme is at most $283n^{3/2-\alpha}/4$ if n^α processors are used. In [7], a parallel scheme for factoring the matrix A on a multiprocessor system is given that is analogous to the wrap around assignment scheme described in the Section 2.5 for dense matrices. This scheme has the property of distributing the work evenly among the processors. The computation time to factor the sparse matrix A on n^α processors with the wrap around scheme is at most $197n^{3/2-\alpha}/4$. However, the data traffic associated with that scheme is less than or equal to $183n^{1+\alpha}/4$. Note that the difference in the computation time with the BLOCC scheme and with the wrap around assignment scheme is less than a factor of two. The BLOCC scheme is able to compute the factor efficiently in the case of the sparse matrices because the processors are now assigned blocks in a wrap around fashion which tends to distribute the load evenly. On the other hand, the data traffic associated with the BLOCC scheme is an order of magnitude less than that for the wrap around assignment scheme. Moreover, in the former scheme, as many as n processors may be used before the total data traffic reaches the maximum value of $O(n^{3/2})$, whereas in the later scheme only up to $n^{1/2}$ processors may be used efficiently. The implications of the reduced data traffic on the performance are as follows.

The sparse BLOCC scheme reduces the communication requirement to $O(n^{1+\alpha/2})$ by removing the constraint of column-level indivisibility. Here the indivisible work unit is the computation corresponding to a nonzero element in the factor. The reduction in the communication requirements is brought about by improving the utilization of the data accessed from shared memory by each processor. Consider the factorization of an $m \times m$ dense matrix. Let the ***data utilization*** of a data element accessed by a processor be defined as the number of computations in which that element is used by that processor divided by m. Since an element in the factor is needed in at most m computations, the maximum utilization of any data accessed is one. Let the ***aggregate data utilization*** for a processor be defined as the average utilization of the individual data elements accessed by that processor. In the BLOCC scheme applied to an $m \times m$ dense matrix, each processor accesses at most $2m^2/\sqrt{p}$ elements from the shared memory and each element is used in at least $m/\sqrt{p}$ computations. Thus, the utilization of each data accessed is at least $1/\sqrt{p}$ and so is the aggregate utilization of all the data accesses. On the other hand, with the column-level work assignment scheme, each processor accesses $O(m^2)$ elements from the shared memory. Of these, only $O(m/p)$ elements have a utilization of one and the data utilization for the remaining elements is $1/p$ which gives an aggregate data utilization of approximately $1/p$. Similar improvements in data utilizations are obtained in factoring a sparse matrix.

It should be noted that the square shape of the submatrix partitions produce the best possible aggregate utilizations. For the algorithm considered here, the data dependencies are such that rectangular and square partitions give rise to high data utilizations. Since the square partitions have the minimum perimeter for a given area, the number of data elements accessed (which is proportional to the perimeter of the partition) for a given work load (which is proportional to the area enclosed), is also a minimum for the square partitions.

An effect of the improvement in the aggregate utilization of data and the resulting reduction in the communication requirements is the segregation of the accesses to the shared data. Since the total data traffic in factoring an $m \times m$ dense matrix using p processors is $O(m^2 \cdot \sqrt{p})$, on an average each processor accesses only $O(m^2/\sqrt{p})$ data. Note that the total shared data is $O(m^2)$. Thus, on an average each element in the shared memory is accessed by $O(\sqrt{p})$ processors. The column-level assignment scheme, however, has a total data traffic of $O(m^2 \cdot p)$ and thus, on an average each processor accesses $O(m^2)$ data or on an average each element in the shared memory is accessed by $O(p)$ processors. An obvious implication of this observation is that for the scheme presented here, not only is the total data traffic reduced but also the requests at individual shared addresses. This can have considerable impact on the performance of the systems with a large number of processors.

As a final remark, note that the data traffic analysis for the sparse BLOCC scheme exploits the fact that the underlying graph satisfies a $\sqrt{n}$-separator theorem. Thus, similar schemes may be developed for any class of graphs satisfying an $f(n)$-separator theorem [13]. In such cases the data dependencies, the fill, and the computation time depend on $f(n)$. In [12] the fill and the bounds on the sequential computation time for various values of $f(n)$ are listed. Here we state the bounds on the corresponding data traffic when the systems are computed using n^{α} processors. The data traffic of factoring a matrix corresponding to an n-vertex 3-dimensional regular grid using n^{α} processors is $O(n^{4/3+\alpha/2})$. For that case the computation time is $O(n^{2-\alpha})$. In general, the total data traffic for computing a factor of a matrix corresponding to a d-dimensional grid is $O(n^{2\sigma+\alpha/2})$, where $\sigma = 1 - 1/d$. The computation cost is $O(n^{3\sigma-\alpha})$. For an n-vertex finite element graph[‡] with no element having more than k boundary vertices, the total data traffic in factoring the associated matrix is $O(k^2 \cdot n^{1+\alpha/2})$. The computation time is $O(k^3 \cdot n^{3/2-\alpha})$. All these quantities are optimal in an order of magnitude sense.

[‡]A *finite element graph* is any graph formed from a planar embedding of a planar graph by adding all possible diagonals to each face; i.e., there is a clique corresponding to each face of the embedded planar graph.

References

[1] P. Charrier and J. Roman. *Study of the Parallelism Induced by a Nested Dissection Method and of its Implementation on a Message Passing Multiprocessor Computer.* Technical Report I-8722, Universite de Bordeaux I, Talence, France, 1987.

[2] J. A. George, M. T. Heath, and J. W. H. Liu. Parallel cholesky factorization on a shared-memory multiprocessor. *Linear Algebra and Its Applications*, 77:165–187, 1986.

[3] J. A. George, M. T. Heath, J. W. H. Liu, and E. Ng. *Solution of Sparse Positive Definite Systems on a Shared-Memory Multiprocessor.* Technical Report ORNL/TM-10260, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1987.

[4] J. A. George, M. T. Heath, J. W. H. Liu, and E. Ng. *Sparse Cholesky Factorization on a Local Memory Multiprocessor.* Technical Report ORNL/TM-9962, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1986.

[5] J. A. George, M. T. Heath, E. Ng, and J. W. H. Liu. Symbolic cholesky factorization on a local-memory multiprocessor. *Parallel Computing*, 5:85–95, 1987.

[6] J. A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Inc., Englewood Cliff, NJ, 1981.

[7] J. A. George, J. W. H. Liu, and E. Ng. Communication reduction in parallel sparse cholesky factorization on a hypercube. In M. T. Heath, editor, *Hypercube Multiprocessors 1987*, pages 576–586, SIAM Publication, 1987.

[8] J. R. Gilbert and R. E. Tarjan. The analysis of a nested dissection algorithm. *Numerical Mathematics*, 50:377–404, 1987.

[9] J. R. Gilbert and E. Zmijewski. *A Parallel Graph Partitioning Algorithm for a Message-Passing Multiprocessor.* Technical Report TR 87-803, Department of Computer Science, Cornell University, Ithaca, NY, 1987.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 1983.

[11] M. T. Heath. *Parallel Cholesky Factorization in Message-Passing Multiprocessor Environments.* Technical Report ORNL-6150, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1985.

[12] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM Journal of Numerical Analysis*, 16:346–358, 1979.

[13] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36:177–189, 1979.

[14] J. W. H. Liu. Computational models and task scheduling for parallel sparse cholesky factorization. *Parallel Computing*, 3:327–342, 1986.

[15] V. K. Naik. *On the Computation and Communication Tradeoffs and their Impact on the Performance of Asynchronous Multiprocessor Systems.* PhD thesis, Computer Science Department, Duke University, Durham, NC., 1988.

[16] V. K. Naik and M. L. Patrick. Analysis of communication requirements of sparse cholesky factorization with nested dissection ordering. In G. M. Rodrigue, editor, *Parallel Processing for Scientific Computation*, chapter 2, pages 9–14, SIAM Publication, 1989.

[17] Y. Saad. Communication complexity of the gaussian elimination algorithm on multiprocessors. *Linear Algebra and Its Applications*, 77:315–340, 1986.

[18] P. H. Worley and R. Schreiber. Nested dissection on a mesh-connected processor array. In A. Wouk, editor, *Proceedings of the ARO Workshop on New Computing Environments: Parallel, Vector, and Systolic*, 1986.

[19] E. Zmijewski and J. R. Gilbert. *A Parallel Algorithm for Large Sparse Symbolic and Numeric Cholesky Factorization on a Multiprocessor.* Technical Report 86-733, Department of Computer Science, Cornell University, Ithaca, NY, 1986.

National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CR-181863 ICASE Report No. 89-40	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Data Traffic Reduction Schemes for Cholesky Factorization on Asynchronous Multiprocessor Systems		5. Report Date June 1989
		6. Performing Organization Code
7. Author(s) Vijay K. Naik Merrell L. Patrick		8. Performing Organization Report No. 89-40
		10. Work Unit No. 505-90-21-01
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225		11. Contract or Grant No. NAS1-18107 NAS1-18605
		13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Richard W. Barnwell

Proc. of 1989 ACM International Conf. on Supercomputing

Final Report

16. Abstract

Communication requirements of Cholesky factorization of dense and sparse symmetric, positive definite matrices are analyzed. The communication requirement is characterized by the data traffic generated on multiprocessor systems with local and shared memory. Lower bound proofs are given to show that when the load is uniformly distributed the data traffic associated with factoring an $n \times n$ dense matrix using n^{α}, $\alpha \leq 2$, processors is $\Omega(n^{2+\alpha/2})$. For an $n \times n$ sparse matrices representing a $\sqrt{n} \times \sqrt{n}$ regular grid graph the data traffic is shown to be $\Omega(n^{1+\alpha/2})$, $\alpha \leq 1$.

Partitioning schemes that are variations of block assignment scheme are described and it is shown that the data traffic generated by these schemes are asymptotically optimal. The schemes allow efficient use of up to $O(n^2)$ processors in the dense case and up to $O(n)$ processors in the sparse case before the total data traffic reaches the maximum value of $O(n^3)$ and $O(n^{3/2})$, respectively. It is shown that the block based partitioning schemes allow a better utilization of the data accessed from shared memory and thus reduce the data traffic than those based on column-wise wrap around assignment schemes.

17. Key Words (Suggested by Author(s)) Cholesky factorization, communication requirements, asynchronous multi-processor systesm. local and shared memory	18. Distribution Statement 59 - Mathematical and Computer Sciences Unclassified - Unlimited

19. Security Classif. (of this report) Unclassifed	20. Security Classif. (of this page) Unclassified	21. No. of pages 31	22. Price A03

NASA FORM 1626 OCT 86