Abstract

We present a column-oriented distributed algorithm for factoring a large sparse symmetric positive definite matrix on a local-memory parallel processor. Processors co-operate in computing each column of the Cholesky factor by calculating independent updates to the corresponding column of the original matrix. These updates are sent in a fan-in manner to the processor assigned to the column, which then completes the computation. Experimental results on an Intel iPSC/2 hypercube demonstrate that the method is effective and achieves good speedups.

A Fan-in Algorithm for Distributed Sparse Numerical Factorization

Cleve Ashcraft,[†] *Stanley C. Eisenstat,*[‡]
and Joseph W. H. Liu[§]

Research Report YALEU/DCS/RR-706
15 May 1989

[†] Department of Computer Science, Yale University, New Haven, Connecticut 06520. This research was supported in part by the Office of Naval Research under contract N00014-86-K-0310 and the National Science Foundation under grant DCR-85-21451.

[‡] Department of Computer Science and Research Center for Scientific Computation, Yale University, New Haven, Connecticut 06520. This research was supported in part by the Office of Naval Research under contract N00014-86-K-0310 and the National Science Foundation under grant DCR-85-21451.

[§] Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant A5509.

1. Introduction

With the advent of local-memory parallel processors, many researchers have considered the problem of solving large sparse linear systems on such architectures [3, 4, 5, 7, 9, 11]. In this paper, we present a new column-oriented distributed algorithm for computing the Cholesky factor of a large sparse symmetric positive definite matrix.

We assume that each processor has been assigned a subset of the columns in the matrix and is responsible for computing the corresponding set of columns in the Cholesky factor. We do not address the problem of how to symmetrically reorder the matrix so as to increase the potential parallelism, nor the problem of how to assign columns to processors so as to balance the workload and reduce communication. See [7] for one approach to these problems.

Processors co-operate in computing each column of the Cholesky factor by calculating independent updates to the corresponding column of the original matrix. These updates are sent in a fan-in manner to the processor assigned to the column, which then completes the computation. Thus we refer to this as a *fan-in* algorithm for distributed sparse numerical factorization.

In Section 2, we briefly review Cholesky factorization by columns and introduce the notion of an aggregate update column. In Section 3, we describe how the fan-in algorithm uses aggregate update columns to compute the columns of the Cholesky factor. In Section 4, we present performance results on an Intel iPSC/2 hypercube. The speedups obtained compare favorably with those reported in [7] and [9].

2. Preliminaries

Let A be an $n \times n$ symmetric positive definite matrix and let L be its Cholesky factor, with entries a_{ij} and l_{ij} respectively. The column-Cholesky method computes L column by column:

Algorithm 1: Column-Cholesky Factorization

for column $j := 1$ **to** n **do**
begin

$$\begin{pmatrix} t_j \\ \vdots \\ t_n \end{pmatrix} := \begin{pmatrix} a_{jj} \\ \vdots \\ a_{nj} \end{pmatrix} - \sum_{k<j} l_{jk} \begin{pmatrix} l_{jk} \\ \vdots \\ l_{nk} \end{pmatrix}$$

$$\begin{pmatrix} l_{jj} \\ \vdots \\ l_{nj} \end{pmatrix} := \frac{1}{\sqrt{t_j}} \begin{pmatrix} t_j \\ \vdots \\ t_n \end{pmatrix}$$

end

Here the temporary vector $(t_j, \cdots, t_n)^T$ is used only for clarity; its storage can overlap completely with that of $(l_{jj}, \cdots, l_{nj})^T$.

This formulation is applicable to both dense and sparse matrices. But in the sparse case, the updates $l_{jk}(l_{jk}, \cdots, l_{nk})^T$ to column j are sparse and come only from those preceding columns k of L for which $l_{jk} \neq 0$. These columns are given precisely by the nonzero structure of row L_{j*}:

$$Struct(L_{j*}) = \{k \mid k < j, \ \ l_{jk} \neq 0\} .$$

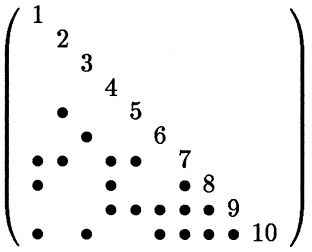

Figure 2.1: A 10-by-10 symmetric matrix (only the lower triangle is shown).

For example, consider the symmetric matrix whose lower triangle is shown in Figure 2.1. Each nonzero in the matrix is represented by a "•", and the matrix has been chosen so that no fill-in occurs during the factorization. The 7-th column of the Cholesky factor is computed as

$$\begin{pmatrix} t_7 \\ t_8 \\ t_9 \\ t_{10} \end{pmatrix} = \begin{pmatrix} a_{77} \\ a_{87} \\ a_{97} \\ a_{10,7} \end{pmatrix} - l_{71} \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix} - l_{72} \begin{pmatrix} \bullet \end{pmatrix} - l_{74} \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix} - l_{75} \begin{pmatrix} \bullet \\ \bullet \end{pmatrix} .$$

Since $Struct(L_{7*}) = \{1, 2, 4, 5\}$, both l_{73} and l_{76} are zero and columns 3 and 6 do not enter into the computation.

To describe the fan-in algorithm for distributed sparse numerical factorization, we now introduce four related notions:

- *factor column* $L_{*j} = (l_{jj}, \cdots, l_{nj})^T$;
- *update column* $l_{jk}(l_{jk}, \cdots, l_{nk})^T$, where $l_{jk} \neq 0$;
- *complete update column* $\sum_{k \in Struct(L_{j*})} l_{jk}(l_{jk}, \cdots, l_{nk})^T$;
- *aggregate update column* $\sum_{k \in K} l_{jk}(l_{jk}, \cdots, l_{nk})^T$, where $K \subseteq Struct(L_{j*})$.

The factor column L_{*j} is an implicit representation of the $n - j$ update columns

$$l_{ij}(l_{ij}, \cdots, l_{nj})^T, \quad i = j + 1, \cdots, n.$$

Update columns and complete update columns are special cases of aggregate update columns where $K = \{k\}$ and $K = Struct(L_{j*})$ respectively.

In Algorithm 1, the column-Cholesky method is formulated in terms of complete update columns — the complete update column for column j is computed and subtracted from $(a_{jj}, \cdots, a_{nj})^T$.

It can also be formulated in terms of update columns — the update columns for column j are computed and subtracted one at a time. This is the approach used in a nodal sparse matrix factorization code, and in the distributed algorithm presented in [7] (where the update columns are sent implicitly as factor columns).

And it can be formulated in terms of aggregate update columns — the update columns for column j are grouped into independent sets and the corresponding aggregate update columns are computed and subtracted one at a time. This is the approach used in a supernodal sparse matrix factorization code [2] (where the groups correspond to supernodes), and is the basis for the fan-in algorithm presented in the next section.

3. A Fan-in Distributed Algorithm

Assume that we are given a mapping of columns to processors, and let $map[k]$ denote the processor assigned to column k. Then we can write the complete update column for column j as a sum of aggregate update columns (each corresponding to a different processor p):

$$\sum_{k \in Struct(L_{j*})} l_{jk}(l_{jk}, \cdots, l_{nk})^T = \sum_p u[j,\ p]$$

where

$$u[j,\ p] \ = \sum_{k \in row[j,p]} l_{jk}(l_{jk}, \cdots, l_{nk})^T,$$

and

$$row[j,p] \ = \ \{k \in Struct(L_{j*}) \mid map[k] = p\}\ .$$

Note that the update columns which appear in $u[j,p]$ all come from factor columns which are mapped to processor p. Thus $u[j,p]$ can be computed without any interprocessor communication. Of course, if $row[j,p] = \emptyset$, then $u[j,p] = 0$ so that

$$\sum_{k \in Struct(L_{j*})} l_{jk}(l_{jk}, \cdots, l_{nk})^T = \sum_{p \ni row[j,p] \neq \emptyset} u[j,\ p]\ .$$

We now state the fan-in algorithm for distributed numerical factorization. This is a simplification of the actual code, but it does capture the essential ideas.

Algorithm 2: Fan-in Distributed Cholesky Factorization

```
mycols = {j | map[j] = myname} ;
for column j := 1 to n do
  if row[j, myname] ≠ ∅  or  j ∈ mycols  then
    begin
      u := 0 ;
      for  k ∈ row[j, myname]  do
          u := u + l_jk(l_jk, ···, l_nk)^T ;
      if  j ∉ mycols  then
          Send aggregate update column u to processor map[j]
      else
        begin
          t := (a_jj, ···, a_nj)^T − u ;
          while  not all contributions have been received  do
              Receive an aggregate update column u for column j
                from another processor and subtract u from t ;
          L_*j := t / √t_j
        end
    end
```

Here *myname* denotes the processor-id of the processor under consideration.

This algorithm works for any mapping of columns to processors. The underlying logic closely resembles that used in the fan-in scheme of Romine and Ortega [10] for the solution

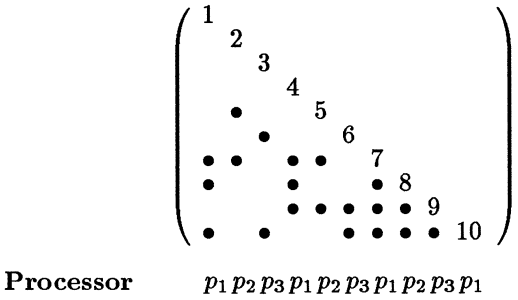

Figure 3.1: Wrap mapping of the columns of a 10-by-10 matrix.

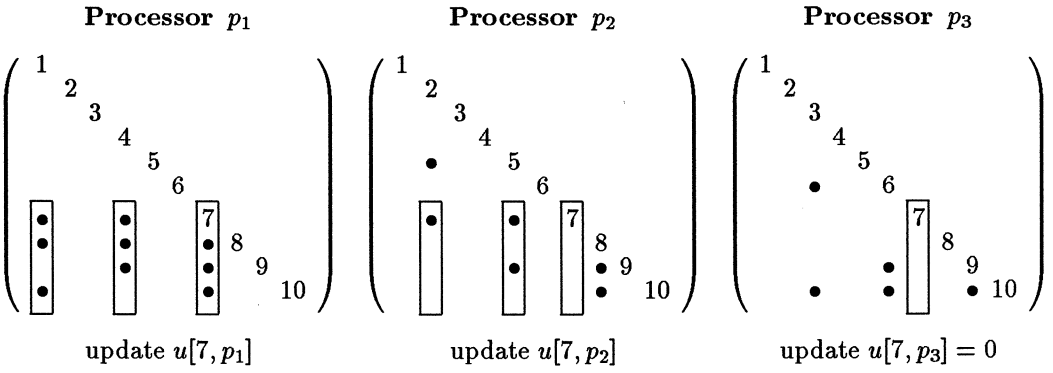

Figure 3.2: Fan-in factorization of column 7.

of dense triangular linear systems.

For simplicity, we have assumed that the aggregate column $u[j,p]$ is sent directly to the destination processor $map[j]$ and that t is formed by subtracting each of these contributions from $(a_{jj},\cdots,a_{nj})^T$. While this approach works irrespective of the connectivity of the multiprocessor network, one could also take advantage of the underlying topology by partially summing the aggregate update columns on their way to the destination processor.

To illustrate the operation of Algorithm 2, we use the example from Section 2. Assume that the columns have been assigned to three processors (p_1, p_2, and p_3) using a simple wrap mapping (see Figure 3.1). The columns assigned to each processor are shown in Figure 3.2.

Column 7 of the Cholesky factor belongs to processor p_1, and there are contributing update columns from processors p_1 and p_2, but not from processor p_3 (since both l_{73} and l_{76} are zero). Processor p_2 has to compute its contribution to column 7

$$u[7,p_2] = l_{72}\begin{pmatrix}\bullet\\ \\ \end{pmatrix} + l_{75}\begin{pmatrix}\bullet\\ \bullet\end{pmatrix},$$

Problem	Serial Time	8 Processors		16 Processors		32 Processors	
		Time	Speedup	Time	Speedup	Time	Speedup
31x31	2.74	0.54	5.07	0.39	7.03	0.32	8.56
63x63	23.26	3.86	6.03	2.41	9.65	1.56	14.91
125x63	60.21	9.26	6.50	5.67	10.62	3.52	17.11

Table 1: Parallel factorization time and speedup on hypercube

and send it to processor p_1. Processor p_1 has to compute its contribution to column 7

$$u[7,p_1] = l_{71} \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix} + l_{74} \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix} ,$$

subtract the two aggregate update columns

$$\begin{pmatrix} t_7 \\ t_8 \\ t_9 \\ t_{10} \end{pmatrix} = \begin{pmatrix} a_{77} \\ a_{87} \\ a_{97} \\ a_{10,7} \end{pmatrix} - u[7,p_1] - u[7,p_2] ,$$

and finally compute the factor column $(l_{7,7}, \cdots, l_{10,7})^T$.

4. Experimental Results

The distributed fan-in algorithm for sparse Cholesky factorization was implemented in C and run on an Intel iPSC/2 hypercube with Weitek 1167 floating-point chips.

The test problems were nine-point finite-difference operators on rectangular grids. We used the nested dissection ordering [6] since it gives optimal-order fill and a well-balanced elimination tree. We used the subtree-to-subcube mapping [8] to assign processors to columns since it gives good load balance and reduced communication. Table 1 contains the timing results and the corresponding speedups for three grid problems. The speedups are relative to a state-of-the-art serial code, again written in C.

Two other approaches to distributed numerical factorization are:

- the *fan-out* algorithm, in which each factor column is sent from its originating processor to every destination processor that needs it [7];
- a distributed version of the *multifrontal* method [9].

The fan-out code achieved a speedup of 5.54 when solving an L-shaped grid problem with 2614 unknowns on a 16-processor hypercube [7]. The multifrontal code achieved a speedup of 9.5 when solving a nine-point problem for a 65×65 grid on a 16-processor hypercube [9]. The speedups given in Table 1 compare favorably with these results and suggest the potential of the new distributed scheme.

However, one should not draw conclusions on the relative merits of these approaches based on these statistics since the problems, machines, and baseline sequential codes differ. A thorough and detailed comparison of the fan-in, fan-out, and distributed multifrontal methods, and their respective implementations, will be given in [1].

Acknowledgement. We would like to thank Andy Sherman for his helpful comments and suggestions.

Note added in proof. The basic fan-in scheme has been devised independently by Earl Zmijewski.

References

[1] C. Ashcraft, S. Eisenstat, J. Liu, and A. Sherman, *A Comparison of Three Distributed Sparse Factorization Schemes.* Presented at the SIAM Symposium on Sparse Matrices, Salishan Resort, Gleneden Beach, Oregon, May 1989.

[2] C. C. Ashcraft, R. G. Grimes, J. G. Lewis, B. W. Peyton, and H. D. Simon, "Progress in sparse matrix methods for large linear systems on vector supercomputers," *International Journal of Supercomputer Applications*, 1(4):10–30, 1987.

[3] R. E. Benner, G. R. Montry, and G. G. Weigand, "Concurrent multifrontal methods: Shared memory, cache, and frontwidth issues," *International Journal of Supercomputer Applications*, 1(3):26–44, 1987.

[4] I. S. Duff, "Parallel implementation of multifrontal schemes," *Parallel Computing*, 3:193–204, 1986.

[5] I. S. Duff, N. I. M. Gould, M. Lescrenier, and J. K. Reid, *The Multifrontal Method in a Parallel Environment*, Technical Report CSS 211, Harwell Laboratory, Oxfordshire, England, 1987.

[6] J. A. George, "Nested dissection of a regular finite element mesh," *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.

[7] J. A. George, M. Heath, J. W. H. Liu, and E. Ng, "Sparse Cholesky factorization on a local-memory multiprocessor," *SIAM Journal on Scientific and Statistical Computing*, 9:327–340, 1988.

[8] 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*, pp. 576–586, SIAM, 1987.

[9] R. Lucas, *Solving Planar Systems of Equations on Distributed-memory Multiprocessor*, PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, California, 1987.

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

[11] E. Zmijewski, *Sparse Cholesky Factorization on a Multiprocessor*, PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, 1987.