The scalability of parallel computers for sparse QR factorisation

David J. Miron*

(Received 7 August 2000)

Abstract

Sparse linear systems occur in areas such as finite element methods and statistics. These systems are often solved on parallel computers due to their size. In this paper a theoretical analysis of parallel sparse QR factorisation using a multifrontal method is undertaken. The analysis is quantified by some estimates of parallel speeds up for various parallel computers. These estimates show that only moderate parallel speedups can be attained.

*CRC for Advanced Computation, Research School of Information Sciences and Engineering, Australian National University, Canberra, ACT 0200, Australia.

[0]See http://anziamj.austms.org.au/V42/CTAC99/Miro for this article and ancillary services, © Austral. Mathematical Soc. 2000. Published 27 Nov 2000.

Contents

1 Introduction

The primary focus of this work is the factorisation $A = QR$, where A is a sparse real $m \times n$ $(m \geq n)$ matrix, Q is orthogonal, R is upper triangular and the number of nonzero entries in any row or column is less than 10%.

The sparse QR factorisation is important as it can be used to solve sparse linear least squares problems. These problems occur in areas such as geodesy [7], data mining [9], animal breeding [8] and finite element problems.

A sparse matrix is factorised by storing and operating only on the nonzeroes in the matrix, thereby realising significant savings in storage requirements and factorisation time. However, sparse matrices can be so large that factorisation on single processor computers is not practical, therefore requiring the use of parallel computers.

Pozo [10] provides a theoretical study of the distributed memory multifrontal LU factorisation. His analysis is based upon *completion time*, being the time taken by the *multifrontal method* [3] to perform the numeric factorisation of a sparse matrix. Using some example matrices from the Harwell Boeing Collection [2], Pozo was able to give estimates for the completion time and parallel speedup of various parallel distributed memory computers.

Rather than using example matrices this work uses a model problem to provide a theoretical analysis for the parallel sparse QR factorisation. The model problem is based upon a special case of a *relaxed supernodal elimination tree* [1, 3]. Using the model problem an equation can be derived that gives the theoretical completion time for parallel sparse QR factorisation using a multifrontal method. Having the theoretical completion time allows the calculation of speedups achievable by a parallel computer to be made.

In Section 2, a brief overview of multifrontal methods is given before proceeding to define the model problem in Section 3. Equations for the theoretical completion times are then presented in Section 4 with results given in Section 5. In Section 6 some conclusions are presented.

2 Multifrontal Methods

The multifrontal method of Duff and Reid [3] reduces the factorisation of a sparse matrix to the factorisation of a sequence of small submatrices called *frontal matrices*. A frontal matrix is associated with each column of the matrix A, with the factorisation of a frontal matrix corresponding to the elimination of a column in A. A factorised frontal matrix is dense and consists of a contribution to R and an *updatematrix*, where the update matrix is made up of the rank -1 updates produced by the frontal matrix factorisation. The updatematrix is stored on a stack to be later retrieved and summed into other frontal matrices.

The order in which frontal matrices are factorised and updatematrices retrieved from the stack is governed by a graph structure called the *elimination tree*. An elimination tree is a tree of n nodes with each node corresponding to a column in the matrix A. The columns of the matrix A corresponding to the leaves of the elimination tree can be eliminated independently. The order in which columns are eliminated is determined by a postorder traversal of the tree.

Frontal matrices can be small, resulting in extensive overheads in stack operations and frontal matrix construction. These overheads can be reduced on vector computers by merging nodes of the elimination tree into “relaxed supernodes”. Thus factorisation of a relaxed supernode involves elimination of all the columns associated with that node from the frontal matrix. Frontal matrices associated with relaxed supernodes explicitly allow zeroes into the factorised frontal matrix [1, 3]. In this work the term elimination tree shall refer to a relaxed supernodal elimination tree.

3 The Model Problem

The model problem is based upon a relaxed supernodal elimination tree with the following structure.

Firstly, the elimination tree T is a balanced binary tree consisting of $l = 1 + \log_2 p$ levels and p leaves. A node $t \in T$ at level i of the elimination tree has a *granularity* $\kappa_t = \frac{\kappa}{\nu^i}$ where κ is the number of columns associated with the root node and $\nu > \sqrt{2}$ is the rate at which the granularity of the nodes in the tree decreases between levels and is a constant.

Secondly, let a frontal matrix $\mathcal{F}^t$ and an updatematrix $\mathcal{U}^t$ of a node t in the elimination tree have magnitudes:

$$|\mathcal{F}^t| = \frac{c_1 \kappa^2}{\nu^{2i}} \tag{1}$$

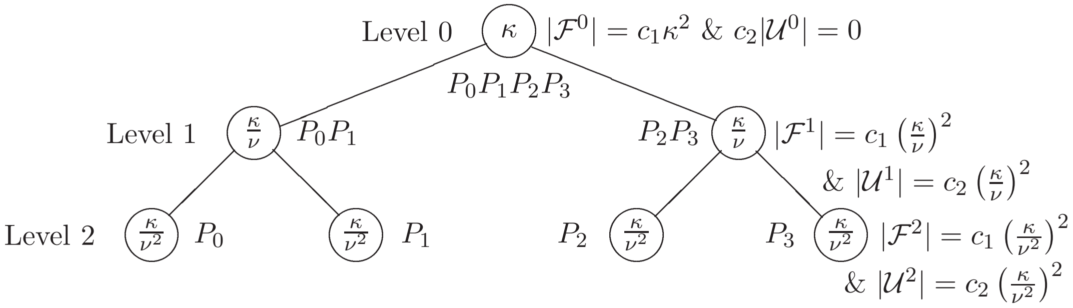
Level 0
κ
$|\mathcal{F}^0| = c_1\kappa^2$ & $c_2|\mathcal{U}^0| = 0$
$P_0P_1P_2P_3$
Level 1
$\frac{\kappa}{\nu}$
P_0P_1
P_2P_3
$|\mathcal{F}^1| = c_1\left(\frac{\kappa}{\nu}\right)^2$
& $|\mathcal{U}^1| = c_2\left(\frac{\kappa}{\nu}\right)^2$
Level 2
$\frac{\kappa}{\nu^2}$
P_0
P_1
P_2
P_3
$|\mathcal{F}^2| = c_1\left(\frac{\kappa}{\nu^2}\right)^2$
& $|\mathcal{U}^2| = c_2\left(\frac{\kappa}{\nu^2}\right)^2$

FIGURE 1: The elimination tree for $p = 4$

and

$$|\mathcal{U}^t| = \begin{cases} 0 & \text{if } i = 0 \\ \frac{c_2\kappa^2}{\nu^{2i}} & \text{if } i > 0 \end{cases} \tag{2}$$

respectively where $|M|$ denotes the number of elements in a matrix M. The parameters c_1 and c_2 are constants and their ratio represents the overdeterminism of the frontal and updatematrices respectively. An example of the elimination tree for the case $p = 4$ is given in Figure 1.

The elimination tree in Figure 1 corresponds to a *separator tree* generated by applying the Recursive Bisection [4] reordering to matrices arising from finite element problems. For finite element problems based on a 2 dimensional grid the granularity of the top level node κ would be proportional to $\sqrt{n}$ [5] while the rate at which the granularity changes between levels ν would be proportional to $\sqrt{2}$.

Figure 1 also shows the processor to elimination tree mapping, the processors being labelled P_0, P_1, P_2 and P_3 respectively. In general terms the parallel factorisation of the elimination tree exploits the natural parallelism in the tree while at the same time distributing the factorisation of a frontal matrix $\mathcal{F}^t$ at level i over 2^{l-i} processors.

For each node $t \in T$ there is a *processor group* and for each processor group there is a *lead processor*. The lead processor is responsible for distributing a frontal matrix $\mathcal{F}^t$ over the other processors within its group for parallel factorisation and the gathering of the resulting R contribution and the updatematrix $\mathcal{U}^t$. On completion of the gathering, the lead processor then sends the updatematrix to the lead processor of its parent node.

4 Completion Times

In this section equations for determining the completion times of the single processor and parallel factorisation of the model problem are derived.

4.1 Single Processor Factorisation

On a single processor the multifrontal QR factorisation involves a postorder traversal of the elimination tree T. On visiting each node $t \in T$ a frontal matrix $\mathcal{F}^t$ is assembled and factorised.

The assembly involves the accumulation of updatematrices $\mathcal{U}^s$ (for which s is a child node of t) into the frontal matrix $\mathcal{F}^t$. For each node $t \in T$ the assembly time is:

$$\mathcal{A}(t) = \gamma\lambda \left(\sum_{s \in \mathrm{child}(t)} |\mathcal{U}^s| \right), \tag{3}$$

where $\lambda \geq 1$ relates to the efficiency of the indirect addressing associated with the assembly process and γ is the average time for one floating point operation (computation cost). In the ensuing discussion it is assumed that the cost of indirect addressing $\lambda = 1$.

After assembly the frontal matrix $\mathcal{F}^t$ is factorised. The factorisation involves the elimination of κ_t columns from the frontal matrix $\mathcal{F}^t$. Using Householder reflections to perform the QR factorisation requires the computation of κ_t Householder vectors and rank 1 updates resulting in at least $4\kappa_t|\mathcal{F}^t|$ operations [6]. This gives an upperbound on the time to factorise a frontal matrix $\mathcal{F}^t$ as:

$$\mathcal{E}(t) = 4\gamma\kappa_t|\mathcal{F}^t|. \tag{4}$$

Thus an upperbound for the completion time for multifrontal QR factorisation on a single processor is the sum of the time required to factor each frontal matrix $\mathcal{F}^t$:

$$\mathcal{T}_{seq} = \sum_{t=1}^{\eta} \left(\mathcal{A}(t) + \mathcal{E}(t)\right), \tag{5}$$

where η is the number of nodes in the elimination tree. Substituting the frontal and updatematrix sizes from (1) and (2) into (5) and simplifying gives a single processor completion time of:

$$\mathcal{T}_{seq} = 4\gamma\kappa^3 c_1 \left[\frac{1}{1-\frac{2}{\nu^3}} \left(1 - \left(\frac{2}{\nu^3}\right)^{l+1} \right) + \frac{c_2}{2\kappa c_1 \left(1-\frac{2}{\nu^2}\right)} \left(1 - \left(\frac{2}{\nu^2}\right)^{l} \right) \right]. \tag{6}$$

4.2 Parallel Factorisation

The time to perform the parallel factorisation is governed by the height of the elimination tree $|h|$ where $|h| = \log_2 p + 1$ and h is the longest path from a leaf node to the root node of the elimination tree.

The parallel factorisation involves communicating updatematrices from child nodes of the elimination tree to their parents. This needs to be done by the lead processor of each child node in the elimination tree.

Let the time for sending k words of data from one processor to another be:

$$M(k) = \alpha + \omega\beta k, \tag{7}$$

where α is the startup cost (latency), β is the per byte transmission time (communication cost) and ω the number of bytes per word. Therefore the cost of communicating an updatematrix is:

$$\mathcal{C}(t) = \alpha + \omega\beta|\mathcal{U}^t|. \tag{8}$$

If it is assumed that the frontal matrix $\mathcal{F}^t$ is on one processor before distribution and gathered back onto that same processor after factorisation, then an upperbound for the distributed factorisation time for a frontal matrix $\mathcal{F}^t$ is:

$$\mathcal{E}(t,p) = 2(p-1)\left(\alpha + \omega\beta\frac{|\mathcal{F}^t|}{p}\right) + \frac{4\gamma\kappa_t|\mathcal{F}^t|}{\phi p}. \tag{9}$$

The first term of (9) reflects the communication cost of distributing and gathering the frontal matrix $\mathcal{F}^t$, while the second term is an upperbound on the time required to factor the frontal matrix $\mathcal{F}^t$ over p processors with the distributed factorisation having a parallel efficiency of $0 < \phi \leq 1$.

Exploiting the natural parallelism of the elimination tree gives the parallel factorisation an optimal total time of:

$$\mathcal{T}_{par} = \min_{\substack{z() \\ 1\leq p\leq|\mathcal{F}^t|}} \max_{h} \left[\mathcal{A}(t) + \mathcal{E}(t,p) + \mathcal{C}(t)\right]. \tag{10}$$

Equation (10) shows that the optimal total time for parallel factorisation, given all possible processor mappings $z()$ and distributing the factorisation of a frontal matrix over an optimal number of processors p, is governed by the longest time taken to traverse the elimination tree from the leaves to the root while performing the factorisation.

Thus the estimated completion time for the parallel factorisation of the model problem is:

$$\begin{aligned}\mathcal{T}^*_{par} = 4\gamma\kappa^3 c_1 \Bigg[& \frac{1}{2^l\phi\left(1-\frac{2}{\nu^3}\right)}\left(1-\left(\frac{2}{\nu^3}\right)^{l+1}\right) + \\ & \frac{\omega\beta}{2\gamma\kappa}\left(\frac{1}{1-\frac{1}{\nu^2}}\left(1-\left(\frac{1}{\nu^2}\right)^{l+1}\right) - \right. \\ & \left.\frac{1}{2^l\left(1-\frac{2}{\nu^2}\right)}\left(1-\left(\frac{2}{\nu^2}\right)^{l+1}\right)\right) + \\ & \frac{c_2}{2\kappa c_1\left(\nu^2-1\right)}\left(1-\left(\frac{1}{\nu^2}\right)^{l}\right)\left(\frac{\omega\beta}{2\gamma}+\nu^2\right) + \\ & \frac{\alpha}{4\gamma\kappa^3 c_1}\left(2^{l+2}-l-6\right)\Bigg] . \end{aligned} \tag{11}$$

Equation (11) is derived from (10) by making the appropriate substitutions for the frontal and updatematrix sizes given by Equations (1) and (2) respectively. The sum for (11) is taken over all levels 0 to l of the elimination tree.

Hence, the speedup for the parallel factorisation is:

$$\mathcal{S} = \frac{\mathcal{T}^*_{seq}}{\mathcal{T}^*_{par}} . \tag{12}$$

TABLE 1: Computer parameters

Computer	γ^{-1} (Mflops)	$\alpha(\mu$ sec)	β^{-1} (MB/sec)	$\frac{\beta}{\gamma}$
Intel iPSC/2 dx3	1.7	322	2.8	0.6
Intel iPSC/860 CX	80.0	322	2.8	28.6
IBM P2SC (Thin node)	810.0	35	110.0	7.4
DEC Alpha 600 EB5 Farm	323.0	500	11.0	29.5
Fujitsu VPP300	2200.0	4	570.0	3.9
Fujitsu AP1000	5.6	7	25.0	0.2

Again assuming that both l and κ are sufficiently large in Equations (6) and (11) then (12) would behave as follows:

$$\mathcal{S}^*_{par} = \phi 2^l = \phi p. \tag{13}$$

The upperbound given in (13) is a theoretical upperbound which in practice would be unreachable.

5 Parallel Speedups

To estimate the scalability of various parallel computers consider the performance figures in Table 1.

TABLE 2: Speedups for the various computers.

Machine	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$
DEC Alpha 600 EB5 Farm	1.83	3.09	4.67	6.21	7.40	8.10
Intel iPSC/860 CX	1.83	3.11	4.72	6.34	7.61	8.43
IBM P2SC (Thin node)	1.95	3.72	6.78	11.46	17.43	23.44
Fujitsu VPP300	1.98	3.85	7.31	13.24	22.23	33.56
Intel iPSC/2 dx3	1.99	3.97	7.87	15.45	29.79	55.51
Fujitsu AP1000	2.00	3.99	7.95	15.76	31.02	60.09

Table 1 gives the computer type, the number of floating point operations per second (γ^{-1}), the latency in micro seconds (α) and the communication speed in megabytes/second (β^{-1}) for the various example parallel computers. It also gives the communication to computation ratio $\frac{\beta}{\gamma}$.

A comparison of the scalability of the example computer architectures using combined parallel factorisation is given in Table 2.

Table 2 gives the speedups $\mathcal{S}$ for the various example computers given in Table 1. The values for the speedup $\mathcal{S}$ were calculated using (12) and setting the frontal matrix constant $c_1 = 2$, the updatematrix constant $c_2 = 1$, the rate that the granularity of the nodes decreases between levels of the elimination tree $\nu = \sqrt[3]{2}$, the parallel efficiency $\phi = 1$ and the granularity of the top level node $\kappa = 1000$. The computers in Table 2 are ranked using the communication to computation ratio $\frac{\beta}{\gamma}$.

Table 2 shows that the Intel iPSC2 scales the best and has the smallest communication to computation ratio. The VPP300 however can achieve only moderate speedups due to its higher communication to computation ratio. This is despite the fact that its completion time would be considerably faster than that of the Intel iPSC2 for a given number of processors. Table 2 also shows that the scalability of a computer is closely linked to the ratio of communication cost to computation cost $\frac{\beta}{\gamma}$ given in Table 1.

In order to understand the interrelationship between the granularity of the top level node of the elimination tree κ and the number of processors p and their effect on the speedup $\mathcal{S}$ a three dimensional plot is given in Figure 2.

Figure 2 plots the speedup $\mathcal{S}$ against the granularity of the top level node κ of the elimination tree and the number of levels $l = \log_2 p$ of the separator tree for the Fujitsu VPP300. Figure 2 was plotted using (12) and setting the frontal matrix constant $c_1 = 2$, the updatematrix constant $c_2 = 1$, the parallel efficiency $\phi = 1$ and the rate of granularity decrease between the levels of the elimination tree $\nu = \sqrt[3]{2}$. The effect of communication is evident in Figure 2 by the crest in the surface.

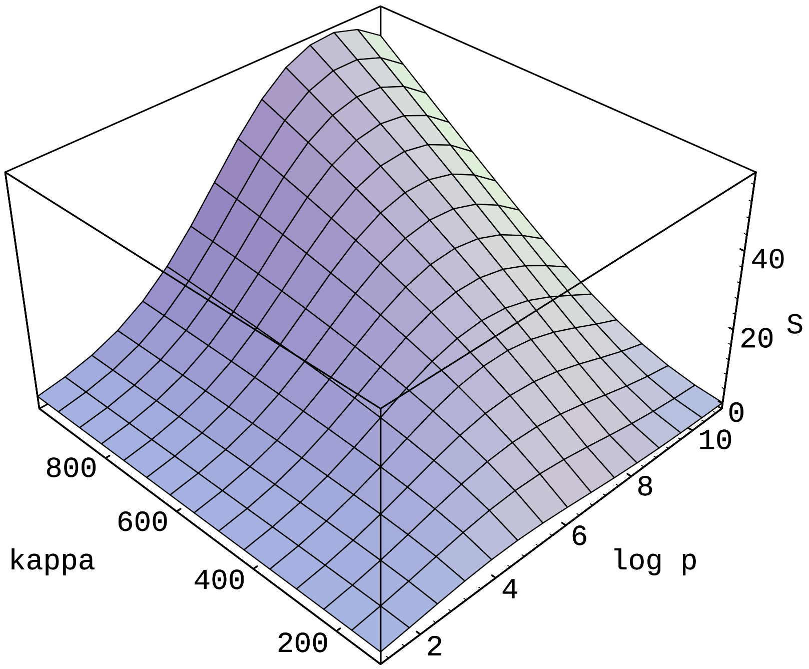

Figure 2: A plot of speedup $\mathcal{S}$ against l and κ

.

6 Conclusions

A model problem based on the separator tree produced by the Recursive Bisection reordering was used to calculate the speedup of various parallel computers.

The parallel factorisation scales linearly with the number of processors. The scalability is closely related to the communication to computation ratio $\frac{\beta}{\gamma}$. The higher this ratio the less scalable the computer architecture.

The analysis also highlights the importance of efficient dense factorisation of frontal matrices. If the dense factorisation is inefficient then the ratio of communication cost to computation cost is reduced giving inflated speedups and larger completion times.

The model problem reflects an ideal situation and no mention has been made of how imbalance in the elimination tree would affect the completion times and speedups. Pozo [10] suggested that such imbalance would have a detrimental effect on parallel performance.

References

[1] C. Ashcraft and R. Grimes. The influence of relaxed supernode partitions on the multifrontal method. *ACM Transactions on Mathematical Software*, 15(4):291–309, 1989. C980, C982

[2] I. Duff. Harwell Boeing sparse matrix collection release I, 1992. C980

[3] I. Duff and J. Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Transactions on Mathematical Software*, 9:302–325, 1983. C980, C980, C981, C982

[4] A. George. Nested disection of a regular finite element mesh. *SIAM J. Numer. Anal.*, pages 345–363, 1973. C983

[5] J. Gilbert and R. Tarjan. The analysis of a nested disection algorithm. *Numerische Mathematik*, 50:377–404, 1987. C983

[6] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 2nd edition, 1989. C985

[7] G. Golub and R. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra and its Applications*, 34:3–27, 1980. C980

[8] M. Hegland. On the computation of breeding values. In *Proceedings of the CONPAR-90-VAPPIV, Joint International Conference on Vector and Parallel Processing*, pages 232–242, Zurich, 1990. C980

[9] M. Hegland. Personal communication, February 1997. C980

[10] R. Pozo. Performance modeling of sparse matrix methods for distributed memory architectures. In *Proceedings of Parallel Processing: CONPAR 92 - VAPP V*, Lecture Notes in Computer Science No. 634, pages 677–688. Springer-Verlag, 1992. C980, C993