Pergamon

Computers Math. Applic. Vol. 30, No. 2, pp. 49–69, 1995
Copyright©1995 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0898-1221/95 \$9.50 + 0.00

0898-1221(95)00077-1

Parallelizing Strassen's Method for Matrix Multiplication on Distributed-Memory MIMD Architectures

C.-C. Chou, Y.-F. Deng,* G. Li and Y. Wang
Center for Scientific Computing, The University at Stony Brook
Stony Brook, NY 11794-3600, U.S.A.

Dedicated to Professor James G. Glimm on the occasion of his 60th birthday

(Received and accepted October 1994)

Abstract—We present a parallel method for matrix multiplication on distributed-memory MIMD architectures based on Strassen's method. Our timing tests, performed on a 56-node Intel Paragon, demonstrate the realization of the potential of the Strassen's method with a complexity of $4.7M^{2.807}$ at the system level rather than the node level at which several earlier works have been focused. The parallel efficiency is nearly perfect when the processor number is the power of 7. The parallelized Strassen's method seems always faster than the traditional matrix multiplication methods whose complexity is $2M^3$ coupled with the BMR method and the Ring method at the system level. The speed gain depends on matrix order M: 20% for $M \approx 1000$ and more than 100% for $M \approx 5000$.

Keywords—Matrix multiplication, Parallel computation, Strassen's method.

1. INTRODUCTION

As the heart of many linear algebra algorithms, matrix multiplication (MM) has been made more and more efficient during the past decades. The complexity of MM for a matrix of order M has dropped [1,2] from $O(M^3)$ for the traditional method (referred to as the T-method, hereafter), to $O(M^{2.807})$ for Strassen's method [3] (referred to as the S-method, hereafter), to $O(M^{2.376})$ for the Winograd method (a.k.a., Coppersmith-Winograd method) [4,5]. This complexity reduction gives rise to many efficient algorithms in other areas of linear algebra: inverting a matrix, solving a system of linear equations, computing the eigenvalues of a matrix [6], and calculating the determinant of a matrix, etc.

Of course, the full potential of these efficient methods can be realized only on large matrices, which require large machines such as parallel computers. Thus, designing efficient parallel algorithms for these methods becomes essential. The parallelization of the general linear algebra routines on distributed-memory MIMD architectures has achieved reasonable success [7]. But, due to the complication of these dedicated MM methods, the progress has been relatively slower. In fact, Manber [8] in 1989 claimed that S-method cannot be easily parallelized. In addition, the Winograd method has not been parallelized so far. Indeed, the attempts for the parallelization

This project was initiated after a fruitful conversation of Y.-F.D. with S.-T. Yau of Harvard University, in the summer of 1993 in Hong Kong. Y.-F.D. thanks D. Scott and W.-L. Wang of Intel Supercomputer Systems Division for discussions on local Strassen implementation, and D. Trystram who brought their unpublished work to our attention, and C.-C.C. thanks S. Huss-Lederman on their BMR implementation on Intel Delta machine. All four of us are supported by the Army Research Office, Grant DAAL03-92-G-0185 and through the Mathematical Sciences Institute of Cornell University under subcontract to the University at Stony Brook, ARO contract number DAAL03-91-C-0027, and the National Science Foundation, Grant DMS-9201581.

*Author to whom all correspondence should be sent. Email address: Yuefan.Deng@sunysb.edu

Table 1. The methods used in our test and the naming convention.

Local\Global	S-Method	BMR	Ring
Low-level coding	SL	BL	RL
High-level coding	SH	BH	RH

of the S-method have been mainly focused on shared-memory machines, such as Bailey's work [9] done on the Cray-2 and Cray Y-MP with no more than 8 processors, and the data-parallel SIMD architectures, such as work [10] done on the MasPar MP-1. Some distributed-memory MIMD system vendors, such as Intel for its Paragon [11], supply node-level library routines that use the S-method. There is, thus far, no reported work[1] on parallelizing the S-method on message-passing parallel systems at the *system* level. However, there are many efforts and resulting routines for parallelizing the T-method. The report, PUMMA: Parallel Universal Matrix Multiplication Algorithms [12], documented three methods including the block scattered decomposition, a variation of the Broadcast-Multiply-Roll (BMR) method [13,14]. A Ring method, another variation of the BMR method, is also used to parallelize the T-method.

In this paper, we introduce a method to parallelize the S-method. For a comparison, we study six different implementations obtained by mixing three global methods (the S-method, BMR method, and the Ring method at the system level) and two implementations locally at the node level by using the T-method (one implementation uses the routine *sgemm* [15] coded in i860 assembler by Intel and the other is a home-grown routine for matrix multiplication coded in Fortran.) The assembler-coded methods are denoted as L-method while the Fortran-coded methods are denoted as H-method. Thus, these six methods are named as SH (global S-method and local H-method), SL (global S-method and local L-method), BH, BL, RH, and RL. By this set of six numerical experiments with various matrix orders and machine sizes, we focus on demonstrating the superiority of S-method at the *system* level.

In Section 2, we describe and analyze the performance of the BMR and Ring methods. In Section 3, we introduce our method—the parallel Strassen's method—and describe its implementation. In Section 4, we analyze the performance of the six different methods implemented. Finally, the conclusions and some future generalizations of our method are given in Section 5.

2. BMR AND RING METHODS AND PARALLEL EFFICIENCY

Many have proposed to merge the local S-method with simpler global methods like BMR or Ring method. The advantage of this idea lies in its ease of implementation, while the shortcomings are that:

(1) small node memory limits the strength of the S-method (which is useful only for a large matrix), and
(2) these parallel algorithms do not scale; i.e., the parallel efficiency drops as the number of processors increases.

Now, let us study the parallel efficiency of these two global methods (B- and R-) coupled with S-method as the "core" algorithm, locally on each node.

Let A and B be two $M \times M$ square matrices whose product is another matrix C, and P be the number of processors used to perform the MM. In this section, we briefly describe how to parallelize the T-method and analyze the performance.

2.1. The BMR Method

The BMR method has been discussed extensively in other publications. (See, for example, [13].) In this section, we concentrate on analyzing its efficiency.

[1]We thank D. Trystram who gave us their preprint that has the methods of fast matrix multiplication on MIMD machines. But, our methods are different.

On one processor, the time is

$$T_1(M) = \left(2M^3 - M^2\right) t_{\text{comp}} \approx 2M^3 t_{\text{comp}},$$

where t_{comp} is the time for one floating-point number operation which may be addition or multiplication. This formula can be written in a general form as

$$T_1(M) \approx \alpha M^\beta t_{\text{comp}}, \tag{1}$$

where $\alpha = 2$ and $\beta = 3$ for the T-method.

On P processors, each being given a submatrix of order $m = M/\sqrt{P}$, the time is split into three portions:

(1) the time to broadcast a submatrix to processors in a row [16],

$$T_B = m^2 t_{\text{comm}} + \left(\sqrt{P} - 2\right) t_{\text{start}},$$

where t_{comm} is the time to transfer one floating-point number and t_{start} is the start up time to initiate a pipeline for data transfer, and

(2) the time to perform submatrix multiplication,

$$T_M = \alpha m^\beta t_{\text{comp}},$$

and

(3) the time to roll up submatrices

$$T_R = m^2 t_{\text{comm}}.$$

The total time is

$$T_P(M) = \sqrt{P}\left(T_B + T_M + T_R\right)$$

where $\sqrt{P}$ appears because the "broadcast-multiply-roll" must be repeated $\sqrt{P}$ times to complete one MM.

The parallel efficiency is

$$E_P(M) = \frac{1}{P^{(3-\beta)/2} + \left(\sqrt{P}/\alpha M^\beta\right)\left[P\left(\sqrt{P} - 2\right)\left(t_{\text{start}}/t_{\text{comp}}\right) + 2M^2\left(t_{\text{comm}}/t_{\text{comp}}\right)\right]}.$$

Asymptotically, for large matrices on large parallel systems, the parallel efficiency approaches

$$E_P(M \to \infty) \to \frac{1}{P^{(3-\beta)/2}}.$$

If the "core" algorithm is the T-method, then $\beta = 3$ and we get

$$E_P(M \to \infty) \to 1,$$

which means an efficient parallel algorithm can be derived this way.

But, if the "core" algorithm is the S-method, then $\beta = 2.807$ and we get

$$E_P(M \to \infty) \to \frac{1}{P^{(3-2.807)/2}} < 1.$$

If we choose $P = 49$ (as we explain later), then $E_P = 69\%$. It means the maximum parallel efficiency for BMR method coupled to S-method is 69% for up to 49 processors.

2.2. The Ring Method

We suppose M is divisible by P, and thus we partition $m = M/P$ rows of matrices A and B to each processor, initially. (Note, the divisibility condition can be relaxed with small loss of efficiency and the requirement of square matrix can be removed without any loss of performance.) The natural way to distribute the data is to give the first m rows of A and B to processor 1, the second m rows of A and B to processor $2, \ldots,$ and the last m rows of A and B to processor P. Each processor then slices the two sets of rows, one from A and the other from B, into P blocks of $m \times m$ submatrices. Symbolically,

$$A = \begin{pmatrix} \bar{a}_{11} & \bar{a}_{12} & \cdots & \bar{a}_{1P} \\ \bar{a}_{21} & \bar{a}_{22} & \cdots & \bar{a}_{2P} \\ \cdots & \cdots & \cdots & \cdots \\ \bar{a}_{P1} & \bar{a}_{P2} & \cdots & \bar{a}_{PP} \end{pmatrix}, \quad B = \begin{pmatrix} \bar{b}_{11} & \bar{b}_{12} & \cdots & \bar{b}_{1P} \\ \bar{b}_{21} & \bar{b}_{22} & \cdots & \bar{b}_{2P} \\ \cdots & \cdots & \cdots & \cdots \\ \bar{b}_{P1} & \bar{b}_{P2} & \cdots & \bar{b}_{PP} \end{pmatrix}.$$

This distribution scheme determines that processor 1 is responsible for producing the first m rows of the resulting matrix C, processors 2 for the second, $\ldots$, and the last processor for the last rows. At step I, all P processors start multiplying the diagonal submatrices of A with their own partition of submatrices of B. At the end of this step, processor 1 has created one term for each submatrix in the first m rows, $\bar{a}_{11}\bar{b}_{11}$, $\bar{a}_{11}\bar{b}_{12}, \ldots,$ $\bar{a}_{11}\bar{b}_{1P}$. It is similar for the other $P-1$ processors. At step II, all processors shift their submatrices left by one submatrix. (For example, processor 1 moves the submatrix $\bar{a}_{12}$ to the position of $\bar{a}_{11}$, processor 2 moves the submatrix $\bar{a}_{23}$ to the position of $\bar{a}_{22}$, and so on.) Of course, this shift is done only within individual processors and there is no need for interprocessor communication. However, at the same step, all P processors must roll up the entire submatrices of B to one row higher in the matrix diagram (e.g., to move submatrices $\bar{b}_{21}, \bar{b}_{22}, \ldots, \bar{b}_{2P}$, to the positions of submatrices $\bar{b}_{11}, \bar{b}_{12}, \ldots, \bar{b}_{1P}$, respectively.) The processor 1 will roll "up" its submatrices to processor P. We repeat steps I and II until all rows of matrix B reach all processors in the fashion of ring and all submatrices in A become diagonal submatrices once.

Now, we study its efficiency. Similarly, on one processor the time is

$$T_1(M) \approx \alpha M^{\beta} t_{\text{comp}}, \tag{2}$$

where $\alpha = 2$ and $\beta = 3$ for the T-method.

On P processors, the time consists of two parts: T_{roll}, time required to move rows of submatrices from one processor to another, and T_{comp}, time needed to perform submatrix multiplications. Thus,

$$T_{\text{roll}} = (P-1)\frac{M^2}{P} t_{\text{comm}} \approx M^2 t_{\text{comm}}$$

and using equation (1), we get

$$T_{\text{comp}} \approx P^2 \alpha \left(\frac{M}{P}\right)^{\beta} t_{\text{comp}}.$$

Therefore, the total time is

$$T_P(M) = T_{\text{roll}} + T_{\text{comp}} \approx M^2 t_{\text{comm}} + P^2 \alpha \left(\frac{M}{P}\right)^{\beta} t_{\text{comp}}.$$

The parallel efficiency becomes

$$E_P(M) = \frac{S_P(M)}{P} = \frac{1}{P^{3-\beta} + \left(P/\alpha M^{\beta-2}\right)\left(t_{\text{comm}}/t_{\text{comp}}\right)}. \tag{3}$$

If we fix P and increase M,

$$E_P(M \to \infty) \to \frac{1}{P^{3-\beta}}.$$

For the T-method where $\beta = 3$, then

$$E_P(M) = \frac{1}{1 + (P/2M)\,(t_{\text{comm}}/t_{\text{comp}})} \to 1$$

as long as $m = M/P$ is kept large enough. Therefore, the Ring method, if used for realistic problems in which m is always large, is well parallelized, but rooted in the T-method with an $O(M^3)$ complexity.

For the S-method where $\beta = 2.807$ and if, again, we choose $P = 49$, $E_P(M \to \infty) \to 0.47$. It means that the best parallel efficiency is only 47%, which is low. If we increase P, E_P becomes even lower. For additional comparison, if we combine the Ring method with the Winograd method in which $\beta = 2.376$, we get $E_P(M \to \infty) \to 0.09$, and the situation is much worse.

In summary, we have the following observations:

(1) the S-method is superior to the T-method in performance, for large matrices at node-level,

(2) as a parallel method, the BMR method is alway superior to the Ring method,

(3) coupling global BMR method or global Ring method with the local T-method can produce parallel-efficient algorithms, but overall performance is low due to low node performance using the T-method, and

(4) coupling global BMR method or global Ring method with the local S-method cannot produce parallel-efficient algorithms.

Approaches (3) and (4) are not desirable.

Obviously, the ideal combination is a global S-method coupled with a local S-method. For ease of comparison, we design a scheme to use S-method at the system level and the T-method (for practical reason, one can easily use the S-method as well, the difference is small) at the node level.

3. STRASSEN'S METHOD

We first briefly describe the conventional S-method in Section 3.1 and then introduce our parallelization of the S-method in Section 3.2. Finally, in Section 3.3, we discuss the implementation of our method.

3.1. Serial

Let A and B be matrices of order $M = m2^{k+1}$ and let C be their product

$$A = \begin{pmatrix} \overline{A}_{11} & \overline{A}_{12} \\ \overline{A}_{21} & \overline{A}_{22} \end{pmatrix}, \quad B = \begin{pmatrix} \overline{B}_{11} & \overline{B}_{12} \\ \overline{B}_{21} & \overline{B}_{22} \end{pmatrix}, \quad C = \begin{pmatrix} \overline{C}_{11} & \overline{C}_{12} \\ \overline{C}_{21} & \overline{C}_{22} \end{pmatrix},$$

where $\overline{A}_{ik}$, $\overline{B}_{ik}$, $\overline{C}_{ik}$ are submatrices of order $M_1 = m2^k$. The pattern goes as follows:

$$\begin{array}{lll}
S_1 = \overline{A}_{11} + \overline{A}_{22}, & P_1 = S_1 S_2, & T_1 = P_1 + P_4, \\
S_2 = \overline{B}_{11} + \overline{B}_{22}, & P_2 = S_3 \overline{B}_{11}, & T_2 = P_7 - P_5, \\
S_3 = \overline{A}_{21} + \overline{A}_{22}, & P_3 = \overline{A}_{11} S_4, & T_3 = P_1 + P_3, \\
S_4 = \overline{B}_{12} - \overline{B}_{22}, & P_4 = \overline{A}_{22} S_5, & T_4 = P_6 - P_2, \\
S_5 = \overline{B}_{21} - \overline{B}_{11}, & P_5 = S_6 \overline{B}_{22}, & \overline{C}_{11} = T_1 + T_2, \\
S_6 = \overline{A}_{11} + \overline{A}_{12}, & P_6 = S_7 S_8, & \overline{C}_{12} = P_3 + P_5, \\
S_7 = \overline{A}_{21} - \overline{A}_{11}, & P_7 = S_9 S_{10}, & \overline{C}_{21} = P_2 + P_4, \\
S_8 = \overline{B}_{11} + \overline{B}_{12}, & & \overline{C}_{22} = T_3 + T_4. \\
S_9 = \overline{A}_{12} - \overline{A}_{22}, & & \\
S_{10} = \overline{B}_{21} + \overline{B}_{22}, & &
\end{array}$$

We have 18 matrix additions and subtractions and 7 MMs in the above formulae. (For a comparison, the T-method requires 8 MMs.) Obviously, the main cost is not due to the matrix addition. It is the MM which is expensive. The multiplications of these submatrices of order M_1 can then be done by the above method recursively. This procedure of MM is called Strassen's method, and it requires [3] $T_{\text{Strassen}} = \alpha M^\beta$ arithmetic operations to multiply a pair of square matrices of order M where $\alpha = 4.7$ and $\beta = \log_2 7 = 2.807$, asymptotically.

3.2. Parallel

First, we decompose the matrix A into 2×2 blocks of submatrices $\overline{A}_{ij}$ where $i, j = 1, 2$. Second, we further decompose these four submatrices into four 2×2 (i.e., 4×4) blocks of submatrices $\overline{a}_{ij}$ $\forall(i,j) \le 4$. We have

$$A = \begin{pmatrix} \overline{A}_{11} & \overline{A}_{12} \\ \overline{A}_{21} & \overline{A}_{22} \end{pmatrix} = \begin{pmatrix} \overline{a}_{11} & \overline{a}_{12} & \overline{a}_{13} & \overline{a}_{14} \\ \overline{a}_{21} & \overline{a}_{22} & \overline{a}_{23} & \overline{a}_{24} \\ \overline{a}_{31} & \overline{a}_{32} & \overline{a}_{33} & \overline{a}_{34} \\ \overline{a}_{41} & \overline{a}_{42} & \overline{a}_{43} & \overline{a}_{44} \end{pmatrix}.$$

Similarly, we perform the same decomposition on matrix B and get

$$B = \begin{pmatrix} \overline{B}_{11} & \overline{B}_{12} \\ \overline{B}_{21} & \overline{B}_{22} \end{pmatrix} = \begin{pmatrix} \overline{b}_{11} & \overline{b}_{12} & \overline{b}_{13} & \overline{b}_{14} \\ \overline{b}_{21} & \overline{b}_{22} & \overline{b}_{23} & \overline{b}_{24} \\ \overline{b}_{31} & \overline{b}_{32} & \overline{b}_{33} & \overline{b}_{34} \\ \overline{b}_{41} & \overline{b}_{42} & \overline{b}_{43} & \overline{b}_{44} \end{pmatrix}.$$

Now, we use the S-method to multiply the matrices A and B, and get the following seven MM expressions,

$$\begin{aligned}
P_1 &= \left(\overline{A}_{11} + \overline{A}_{22}\right)\left(\overline{B}_{11} + \overline{B}_{22}\right), \\
P_2 &= \left(\overline{A}_{21} + \overline{A}_{22}\right)\overline{B}_{11}, \\
P_3 &= \overline{A}_{11}\left(\overline{B}_{12} - \overline{B}_{22}\right), \\
P_4 &= \overline{A}_{22}\left(\overline{B}_{21} - \overline{B}_{11}\right), \\
P_5 &= \left(\overline{A}_{11} + \overline{A}_{12}\right)\overline{B}_{22}, \\
P_6 &= \left(\overline{A}_{21} - \overline{A}_{11}\right)\left(\overline{B}_{11} + \overline{B}_{12}\right), \\
P_7 &= \left(\overline{A}_{12} - \overline{A}_{22}\right)\left(\overline{B}_{21} + \overline{B}_{22}\right).
\end{aligned} \tag{4}$$

Next, we apply the S-method to these seven MMs of the submatrices $\overline{A}$ and $\overline{B}$ so that we obtain 49 MM expressions on submatrices $\overline{a}$ and $\overline{b}$. Taking P_1 as an example, we can expand the one MM on $\overline{A}$ and $\overline{B}$ into seven MMs on $\overline{a}$ and $\overline{b}$ as follows:[2]

$$\begin{aligned}
P_{11} &= \left(\overline{a}_{11} + \overline{a}_{33} + \overline{a}_{22} + \overline{a}_{44}\right)\left(\overline{b}_{11} + \overline{b}_{33} + \overline{b}_{22} + \overline{b}_{44}\right), \\
P_{12} &= \left(\overline{a}_{21} + \overline{a}_{43} + \overline{a}_{22} + \overline{a}_{44}\right)\left(\overline{b}_{11} + \overline{b}_{33}\right), \\
P_{13} &= \left(\overline{a}_{11} + \overline{a}_{33}\right)\left(\overline{b}_{12} + \overline{b}_{34} - \overline{b}_{22} - \overline{b}_{44}\right), \\
P_{14} &= \left(\overline{a}_{22} + \overline{a}_{44}\right)\left(\overline{b}_{21} + \overline{b}_{43} - \overline{b}_{11} - \overline{b}_{33}\right), \\
P_{15} &= \left(\overline{a}_{11} + \overline{a}_{33} + \overline{a}_{12} + \overline{a}_{34}\right)\left(\overline{b}_{22} + \overline{b}_{44}\right), \\
P_{16} &= \left(\overline{a}_{21} + \overline{a}_{43} - \overline{a}_{11} - \overline{a}_{33}\right)\left(\overline{b}_{11} + \overline{b}_{33} + \overline{b}_{12} + \overline{b}_{34}\right), \\
P_{17} &= \left(\overline{a}_{12} + \overline{a}_{34} - \overline{a}_{22} - \overline{a}_{44}\right)\left(\overline{b}_{21} + \overline{b}_{43} + \overline{b}_{22} + \overline{b}_{44}\right).
\end{aligned}$$

Similarly, each of the remaining six MM expressions P_i for $i = 2, 3, \ldots, 7$ can also be expanded into six groups of MMs in terms of $\overline{a}$ and $\overline{b}$.

$$P_{21} = \left(\overline{a}_{31} + \overline{a}_{33} + \overline{a}_{42} + \overline{a}_{44}\right)\left(\overline{b}_{11} + \overline{b}_{22}\right),$$

[2]Obviously, this procedure can be recursively applied to obtain $7^3 = 343$ MMs after decomposing A and B into 8×8 submatrices.

$$P_{22} = (\bar{a}_{41} + \bar{a}_{43} + \bar{a}_{42} + \bar{a}_{44})\bar{b}_{11},$$
$$P_{23} = (\bar{a}_{31} + \bar{a}_{33})\left(\bar{b}_{12} - \bar{b}_{22}\right),$$
$$P_{24} = (\bar{a}_{42} + \bar{a}_{44})\left(\bar{b}_{21} - \bar{b}_{11}\right),$$
$$P_{25} = (\bar{a}_{31} + \bar{a}_{33} + \bar{a}_{32} + \bar{a}_{34})\bar{b}_{22},$$
$$P_{26} = (\bar{a}_{41} + \bar{a}_{43} - \bar{a}_{31} - \bar{a}_{33})\left(\bar{b}_{11} + \bar{b}_{12}\right),$$
$$P_{27} = (\bar{a}_{32} + \bar{a}_{34} - \bar{a}_{42} - \bar{a}_{44})\left(\bar{b}_{21} + \bar{b}_{22}\right);$$

$$P_{31} = (\bar{a}_{11} + \bar{a}_{22})\left(\bar{b}_{13} - \bar{b}_{33} + \bar{b}_{24} - \bar{b}_{44}\right),$$
$$P_{32} = (\bar{a}_{21} + \bar{a}_{22})\left(\bar{b}_{13} - \bar{b}_{33}\right),$$
$$P_{33} = \bar{a}_{11}\left(\bar{b}_{14} - \bar{b}_{34} - \bar{b}_{24} + \bar{b}_{44}\right),$$
$$P_{34} = \bar{a}_{22}\left(\bar{b}_{23} - \bar{b}_{43} - \bar{b}_{13} + \bar{b}_{33}\right),$$
$$P_{35} = (\bar{a}_{11} + \bar{a}_{12})\left(\bar{b}_{24} - \bar{b}_{44}\right),$$
$$P_{36} = (\bar{a}_{21} - \bar{a}_{11})\left(\bar{b}_{13} - \bar{b}_{33} + \bar{b}_{14} - \bar{b}_{34}\right),$$
$$P_{37} = (\bar{a}_{12} - \bar{a}_{22})\left(\bar{b}_{23} - \bar{b}_{43} + \bar{b}_{24} - \bar{b}_{44}\right);$$

$$P_{41} = (\bar{a}_{33} + \bar{a}_{44})\left(\bar{b}_{31} - \bar{b}_{11} + \bar{b}_{42} - \bar{b}_{22}\right),$$
$$P_{42} = (\bar{a}_{43} + \bar{a}_{44})\left(\bar{b}_{31} - \bar{b}_{11}\right),$$
$$P_{43} = \bar{a}_{33}\left(\bar{b}_{32} - \bar{b}_{12} - \bar{b}_{42} + \bar{b}_{22}\right)$$
$$P_{44} = \bar{a}_{44}\left(\bar{b}_{41} - \bar{b}_{21} - \bar{b}_{31} + \bar{b}_{11}\right),$$
$$P_{45} = (\bar{a}_{33} + \bar{a}_{34})\left(\bar{b}_{42} - \bar{b}_{22}\right),$$
$$P_{46} = (\bar{a}_{43} - \bar{a}_{33})\left(\bar{b}_{31} - \bar{b}_{11} + \bar{b}_{32} - \bar{b}_{12}\right),$$
$$P_{47} = (\bar{a}_{34} - \bar{a}_{44})\left(\bar{b}_{41} - \bar{b}_{21} + \bar{b}_{42} - \bar{b}_{22}\right);$$

$$P_{51} = (\bar{a}_{11} + \bar{a}_{13} + \bar{a}_{22} + \bar{a}_{24})\left(\bar{b}_{33} + \bar{b}_{44}\right),$$
$$P_{52} = (\bar{a}_{21} + \bar{a}_{23} + \bar{a}_{22} + \bar{a}_{24})\bar{b}_{33},$$
$$P_{53} = (\bar{a}_{11} + \bar{a}_{13})\left(\bar{b}_{34} - \bar{b}_{44}\right),$$
$$P_{54} = (\bar{a}_{22} + \bar{a}_{24})\left(\bar{b}_{43} - \bar{b}_{33}\right),$$
$$P_{55} = (\bar{a}_{11} + \bar{a}_{13} + \bar{a}_{12} + \bar{a}_{14})\bar{b}_{44},$$
$$P_{56} = (\bar{a}_{21} + \bar{a}_{23} - \bar{a}_{11} - \bar{a}_{13})\left(\bar{b}_{33} + \bar{b}_{34}\right),$$
$$P_{57} = (\bar{a}_{12} + \bar{a}_{14} - \bar{a}_{22} - \bar{a}_{24})\left(\bar{b}_{43} + \bar{b}_{44}\right);$$

$$P_{61} = (\bar{a}_{31} - \bar{a}_{11} + \bar{a}_{42} - \bar{a}_{22})\left(\bar{b}_{11} + \bar{b}_{13} + \bar{b}_{22} + \bar{b}_{24}\right),$$
$$P_{62} = (\bar{a}_{41} - \bar{a}_{21} + \bar{a}_{42} - \bar{a}_{22})\left(\bar{b}_{11} + \bar{b}_{13}\right),$$
$$P_{63} = (\bar{a}_{31} - \bar{a}_{11})\left(\bar{b}_{12} + \bar{b}_{14} - \bar{b}_{22} - \bar{b}_{24}\right),$$
$$P_{64} = (\bar{a}_{42} - \bar{a}_{22})\left(\bar{b}_{21} + \bar{b}_{23} - \bar{b}_{11} - \bar{b}_{13}\right),$$
$$P_{65} = (\bar{a}_{31} - \bar{a}_{11} + \bar{a}_{32} - \bar{a}_{12})\left(\bar{b}_{22} + \bar{b}_{24}\right),$$
$$P_{66} = (\bar{a}_{41} - \bar{a}_{21} - \bar{a}_{31} + \bar{a}_{11})\left(\bar{b}_{11} + \bar{b}_{13} + \bar{b}_{12} + \bar{b}_{14}\right),$$
$$P_{67} = (\bar{a}_{32} - \bar{a}_{12} - \bar{a}_{42} + \bar{a}_{22})\left(\bar{b}_{21} + \bar{b}_{23} + \bar{b}_{22} + \bar{b}_{24}\right);$$

$$P_{71} = (\bar{a}_{13} - \bar{a}_{33} + \bar{a}_{24} - \bar{a}_{44})\left(\bar{b}_{31} + \bar{b}_{33} + \bar{b}_{42} + \bar{b}_{44}\right),$$
$$P_{72} = (\bar{a}_{23} - \bar{a}_{43} + \bar{a}_{24} - \bar{a}_{44})\left(\bar{b}_{31} + \bar{b}_{33}\right),$$
$$P_{73} = (\bar{a}_{13} - \bar{a}_{33})\left(\bar{b}_{32} + \bar{b}_{34} - \bar{b}_{42} - \bar{b}_{44}\right),$$

$$\begin{aligned}
P_{74} &= (\bar{a}_{24} - \bar{a}_{44})\left(\bar{b}_{41} + \bar{b}_{43} - \bar{b}_{31} - \bar{b}_{33}\right),\\
P_{75} &= (\bar{a}_{13} - \bar{a}_{33} + \bar{a}_{14} - \bar{a}_{34})\left(\bar{b}_{42} + \bar{b}_{44}\right),\\
P_{76} &= (\bar{a}_{23} - \bar{a}_{43} - \bar{a}_{13} + \bar{a}_{33})\left(\bar{b}_{31} + \bar{b}_{33} + \bar{b}_{32} + \bar{b}_{34}\right),\\
P_{77} &= (\bar{a}_{14} - \bar{a}_{34} - \bar{a}_{24} + \bar{a}_{44})\left(\bar{b}_{41} + \bar{b}_{43} + \bar{b}_{42} + \bar{b}_{44}\right).
\end{aligned}$$

Therefore, we have identified $7 \times 7 = 49$ MMs and naturally we will either use 7 or 49 processors to perform the MMs. In either case, the MMs distributed to each processor can be performed by the S-method, which leads to a perfect parallelization of the S-method.

After finishing these 49 MMs, we need to combine the resulting P_{ij} $\forall(i,j) \le 7$ to form the final product matrix

$$C = \begin{pmatrix} \overline{C}_{11} & \overline{C}_{12} \\ \overline{C}_{21} & \overline{C}_{22} \end{pmatrix} = \begin{pmatrix} \bar{c}_{11} & \bar{c}_{12} & \bar{c}_{13} & \bar{c}_{14} \\ \bar{c}_{21} & \bar{c}_{22} & \bar{c}_{23} & \bar{c}_{24} \\ \bar{c}_{31} & \bar{c}_{32} & \bar{c}_{33} & \bar{c}_{34} \\ \bar{c}_{41} & \bar{c}_{42} & \bar{c}_{43} & \bar{c}_{44} \end{pmatrix}.$$

First, we define some variables

$$\delta_i = \begin{cases} -1, & \text{if } i = 5;\\ 1, & \text{otherwise.} \end{cases}$$

$$\gamma_i = \begin{cases} -1, & \text{if } i = 2;\\ 1, & \text{otherwise.} \end{cases}$$

$$S_1 = \{1,4,5,7\}, \qquad S_2 = \{2,4\}, \qquad S_3 = \{3,5\}, \quad \text{and} \quad S_4 = \{1,2,3,6\}.$$

The 4×4 blocks of submatrices forming the product matrix C can be written as

$$\begin{aligned}
\bar{c}_{11} &= \sum_{i \in S_1} \delta_i \left(P_{i1} + P_{i4} - P_{i5} + P_{i7}\right),\\
\bar{c}_{12} &= \sum_{i \in S_1} \delta_i \left(P_{i3} + P_{i5}\right),\\
\bar{c}_{13} &= \sum_{i \in S_3} P_{i1} + P_{i4} - P_{i5} + P_{i7},\\
\bar{c}_{14} &= \sum_{i \in S_3} P_{i3} + P_{i5};\\
\bar{c}_{21} &= \sum_{i \in S_1} \delta_i \left(P_{i2} + P_{i4}\right),\\
\bar{c}_{22} &= \sum_{i \in S_1} \delta_i \left(P_{i1} + P_{i3} - P_{i2} + P_{i6}\right),\\
\bar{c}_{23} &= \sum_{i \in S_3} P_{i2} + P_{i4},\\
\bar{c}_{24} &= \sum_{i \in S_3} P_{i1} + P_{i3} - P_{i2} + P_{i6};\\
\bar{c}_{31} &= \sum_{i \in S_2} P_{i1} + P_{i4} - P_{i5} + P_{i7},\\
\bar{c}_{32} &= \sum_{i \in S_2} P_{i3} + P_{i5},\\
\bar{c}_{33} &= \sum_{i \in S_4} \gamma_i \left(P_{i1} + P_{i4} - P_{i5} + P_{i7}\right),\\
\bar{c}_{34} &= \sum_{i \in S_4} \gamma_i \left(P_{i3} + P_{i5}\right);
\end{aligned}$$

$$\begin{aligned}
\bar{c}_{41} &= \sum_{i\in S_2} P_{i2} + P_{i4},\\
\bar{c}_{42} &= \sum_{i\in S_2} P_{i1} + P_{i3} - P_{i2} + P_{i6},\\
\bar{c}_{43} &= \sum_{i\in S_4} \gamma_i \left(P_{i2} + P_{i4}\right),\\
\bar{c}_{44} &= \sum_{i\in S_4} \gamma_i \left(P_{i1} + P_{i3} - P_{i2} + P_{i6}\right).
\end{aligned}$$

We have effectively parallelized the MMs. In fact, we can also group the matrix additions to avoid repetitions in computing matrix sums and in communicating "raw" submatrices. This will be discussed in the next section.

3.3. Implementation

On 7 processors

If we use 7 processors, the implementation is easy. We need to distribute 8 submatrices to 7 processors and each processor will contain no more than three submatrices (the same memory requirement as in the Ring method) so as to minimize the submatrix movement among processors. There are several ways to do this efficiently, and Figure 1 illustrates one simple way for the distribution.

On 49 processors

It is more complex to implement our method on 49 processors. In this case, we need to distribute 32 submatrices to 49 processors with 3 submatrices each. We observe from equation (5) and its expanded 49 MM expressions that submatrices P_1, P_6 and P_7 have the same pattern (needing sums of two submatrices from A and two from B); P_2 and P_4 are similar; P_3 and P_5 are similar. Thus, we divide the 49 MMs into three groups, P_1, P_6 and P_7 in Group 1; P_2 and P_4 in Group 2; P_3 and P_5 in Group 3. Figures 2–4 depict the submatrix distribution and the multiplication processor for the three groups respectively. For Groups 1, 2, and 3, we need 16, 12, and 12 independent submatrices from $\bar{a}$ and $\bar{b}$, respectively. The grouping has another advantage; i.e., no communication occurs among all these groups; every 7 processors form a cluster and there is no need for communication with any of the other $49 - 7 = 42$ processors. In addition, within each group, we perform additions before communication whenever possi.,le. The group size can be further reduced to localize the access of submatrices and therefore reduce the communication.

4. PERFORMANCE COMPARISON

We have conducted timing tests for six methods, SL, SH, BL, BH, RL, and RH discussed above, on a 56-node Paragon, a distributed-memory MIMD parallel computer. For each of these six methods, we collect timing results for 1-, 7-, and 49-processors. The timing results are tabulated in Tables 3–5.

In our tests, we first distribute the submatrix to all participating processors, and each processor then performs its local matrix multiplication with the T-method. All three methods, SL, BL, and RL, share one identical local routine supplied by the vendor (which is tailored to deliver maximum flops from the i860 processor), while the other three methods, SH, BH, and RH, share another identical local routine we created with Fortran. The entire test is done on single precision.

With these data from the three tables (Tables 3–5), we make three sets of plots: Figures 5–7. These figures show the matrix order vs. performance time for processor numbers $P = 1, 7, 49$ in *log-log* plot.

Two points are very clear from examining the figures:

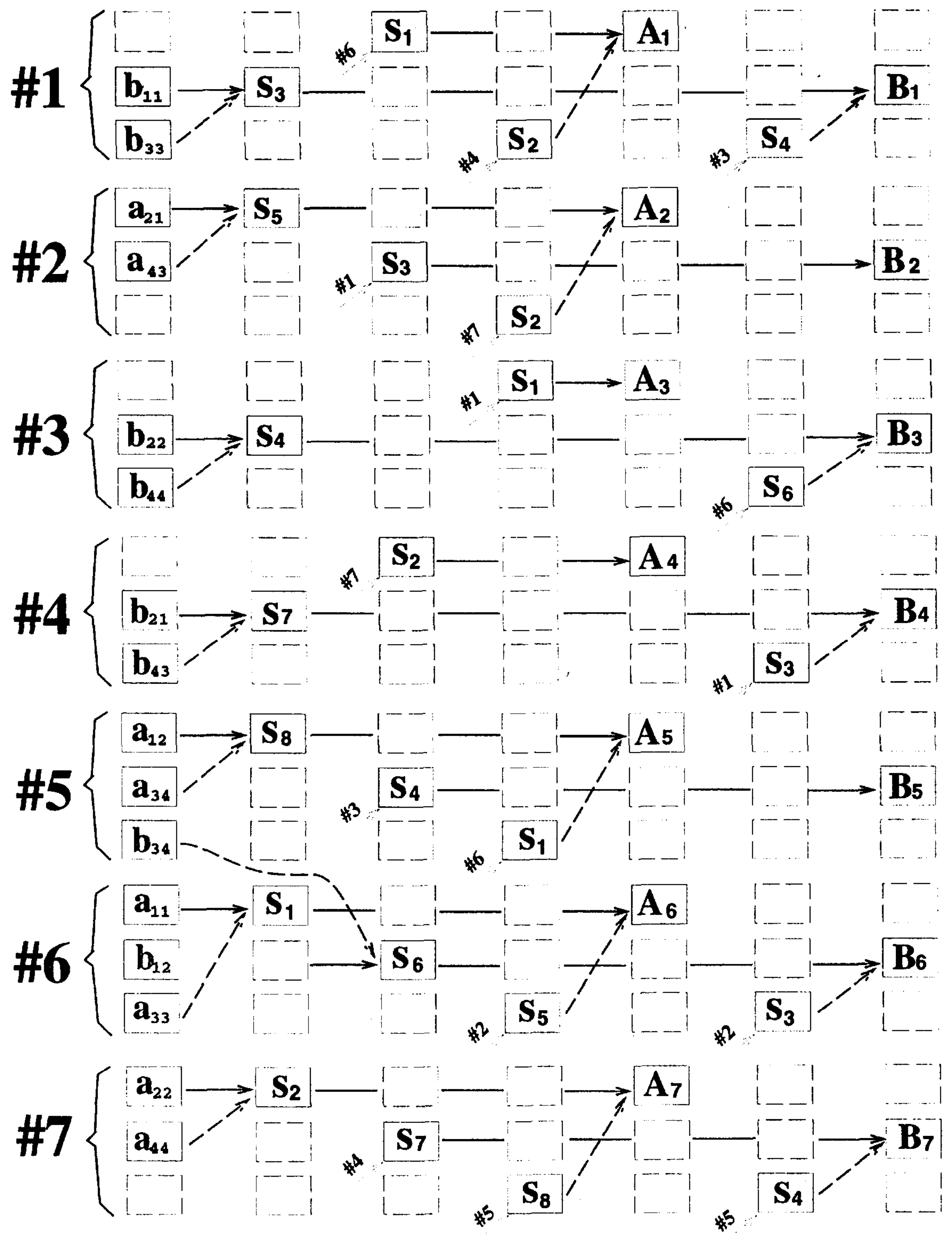


Figure 2. A "map" for the data movements in computing Group 1 submatrices: P_{1i}, P_{6i}, P_{7i} $\forall(i,j) \leq i$. The first column shows the initial submatrix distribution to the processors marked as #1, ..., #7. The second column shows the formation of the intermediate matrices S_i $\forall i \leq 8$ except for $i = 6$, which is done at the third step in which a communication is needed. The communications and calculations are arranged so that only six steps are needed to compute the matrix products. The short arrow with a label like #1 show that the submatrix in the box pointed by the arrow is from processor 1.

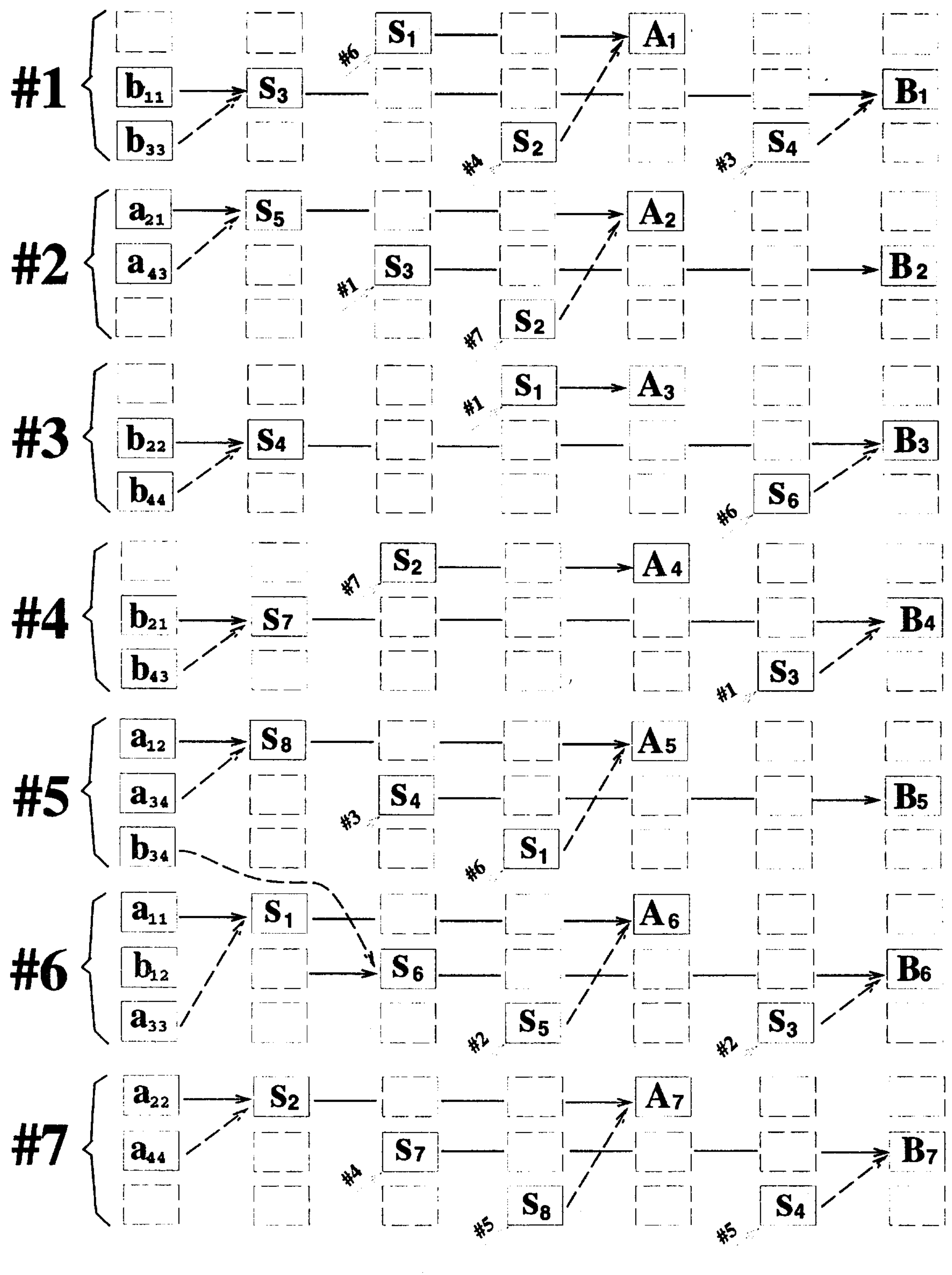


Figure 2. A "map" for the data movements in computing Group 1 submatrices: P_{1i}, P_{6i}, P_{7i} $\forall(i,j) \leq i$. The first column shows the initial submatrix distribution to the processors marked as #1, ..., #7. The second column shows the formation of the intermediate matrices S_i $\forall i \leq 8$ except for $i = 6$, which is done at the third step in which a communication is needed. The communications and calculations are arranged so that only six steps are needed to compute the matrix products. The short arrow with a label like #1 show that the submatrix in the box pointed by the arrow is from processor 1.

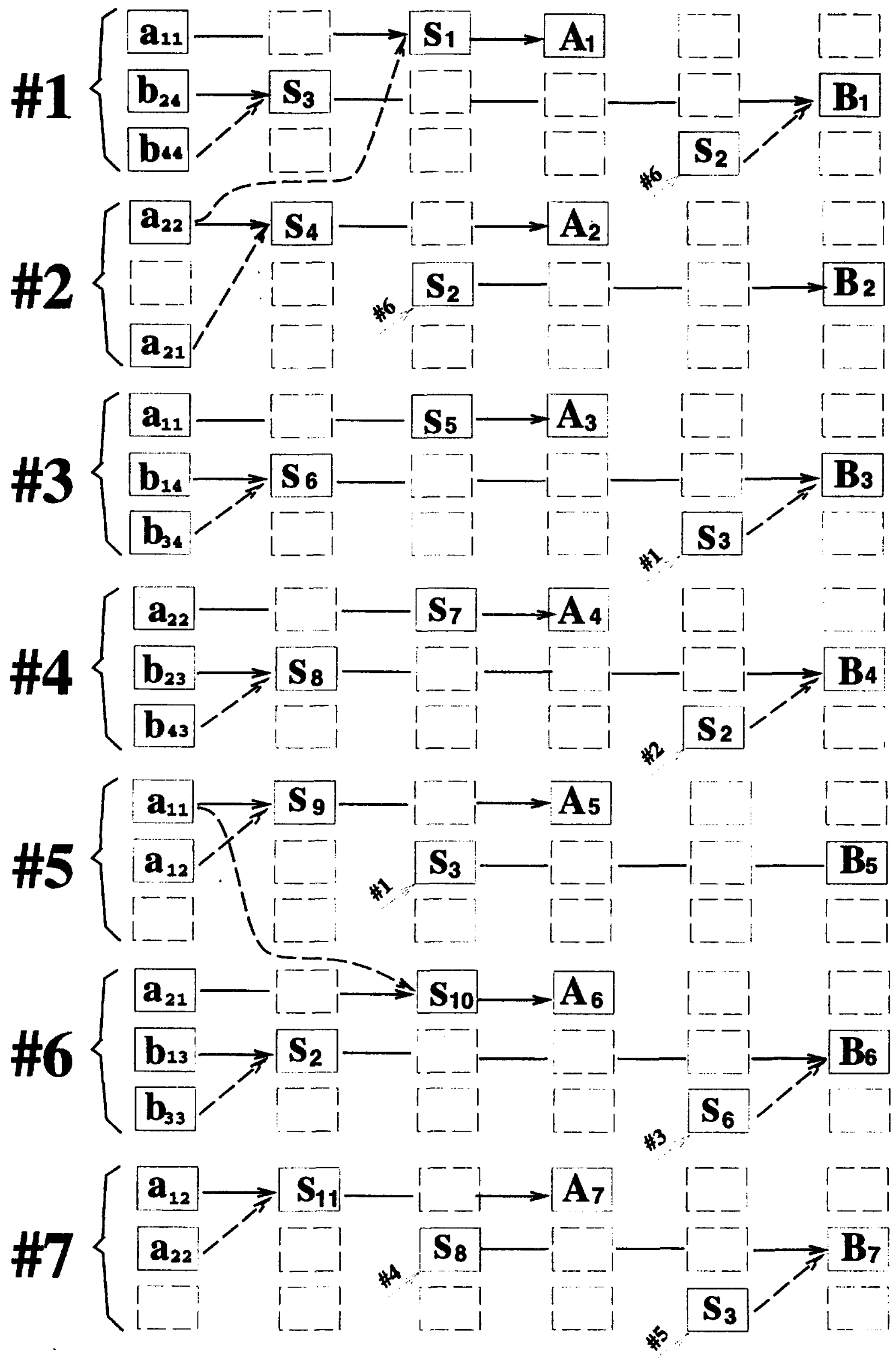
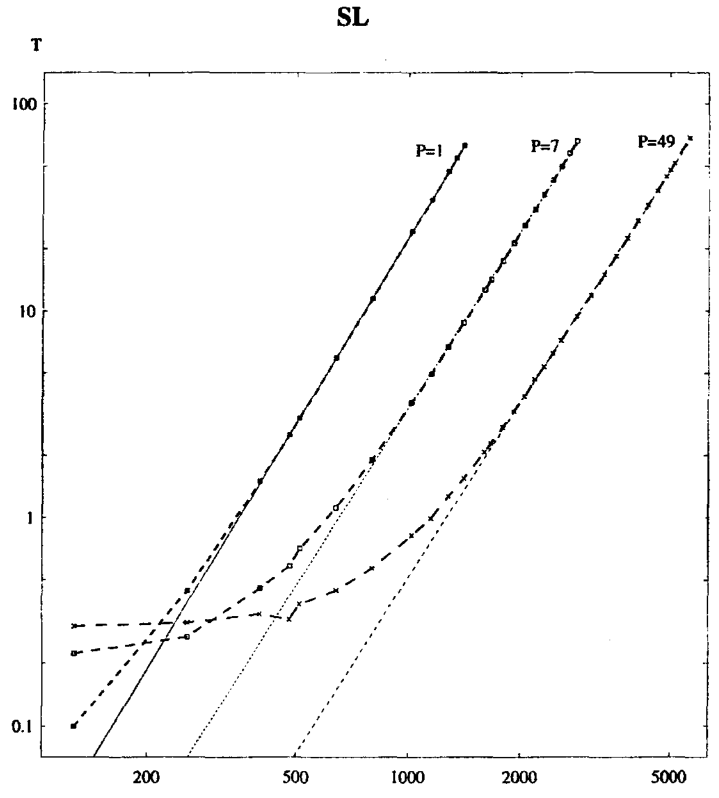


Figure 4. Similar to Figure 2, but for Group 3 submatrices: P_{3i} and P_{5i} $\forall i \leq 7$.



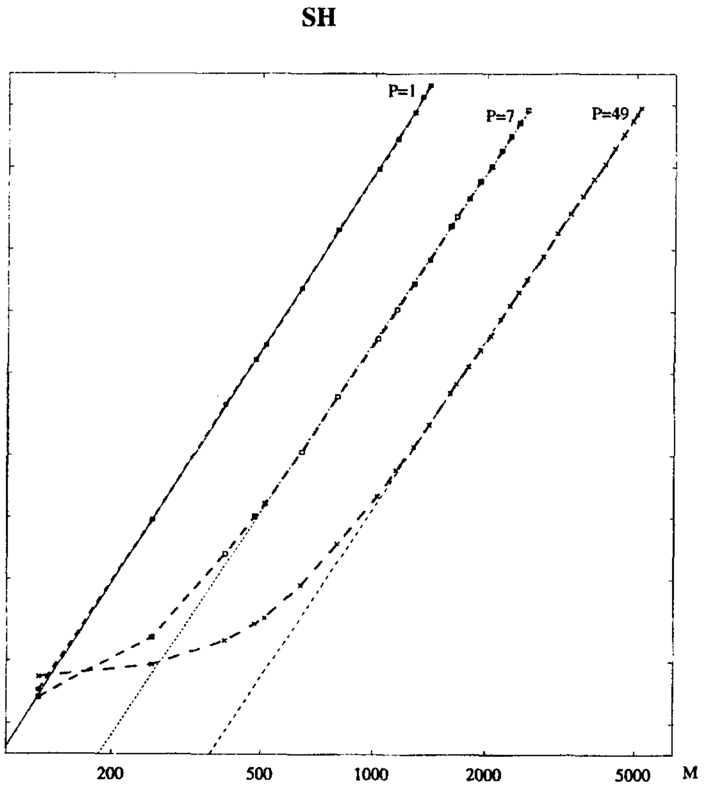
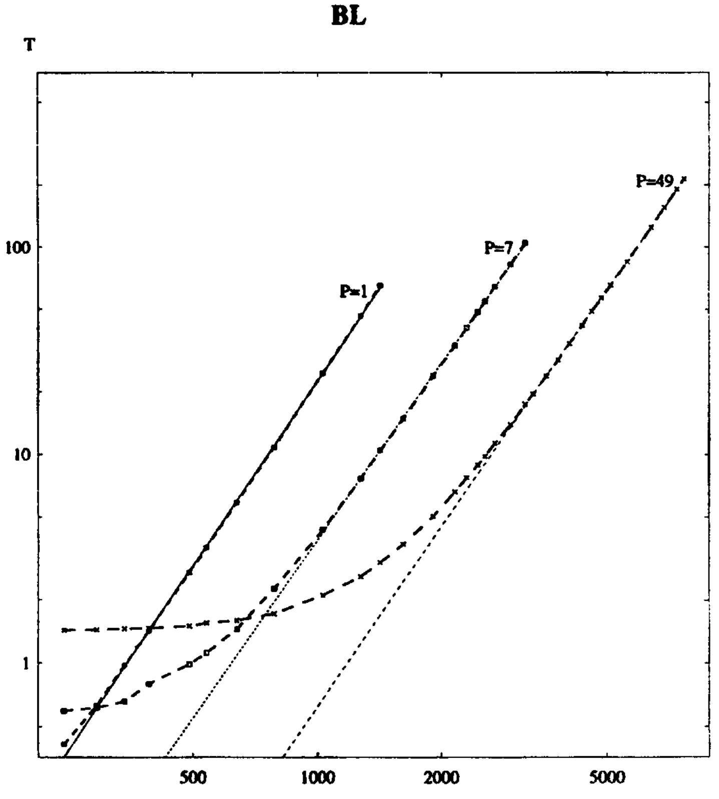


Figure 5. These two graphs show the time (in units of seconds) vs. the matrix order on 1, 7, and 49 processors with the methods of SL and SH. The points (connected by the dotted lines) are the collected data while the straight lines are the fitted results. The figures are a log-log plot.



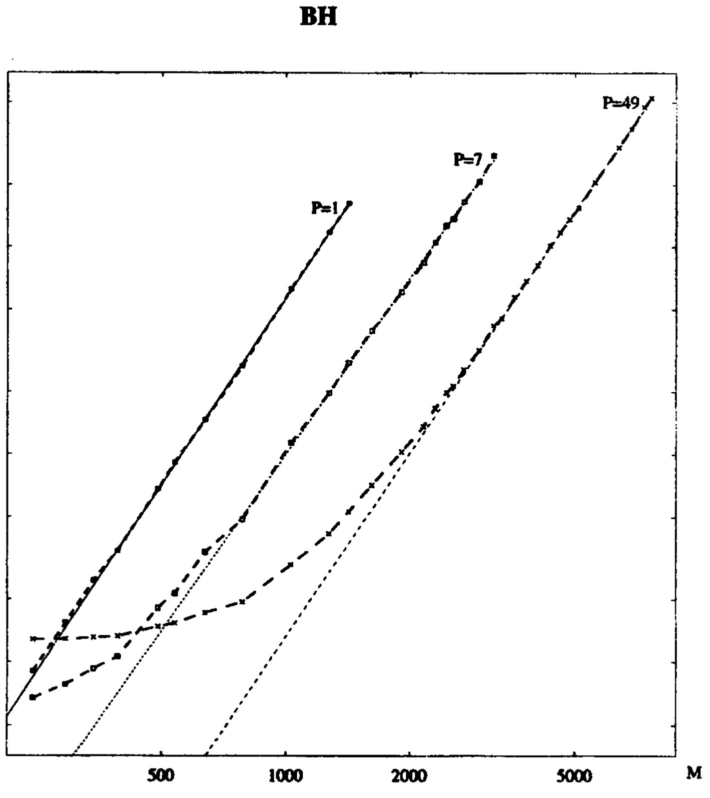
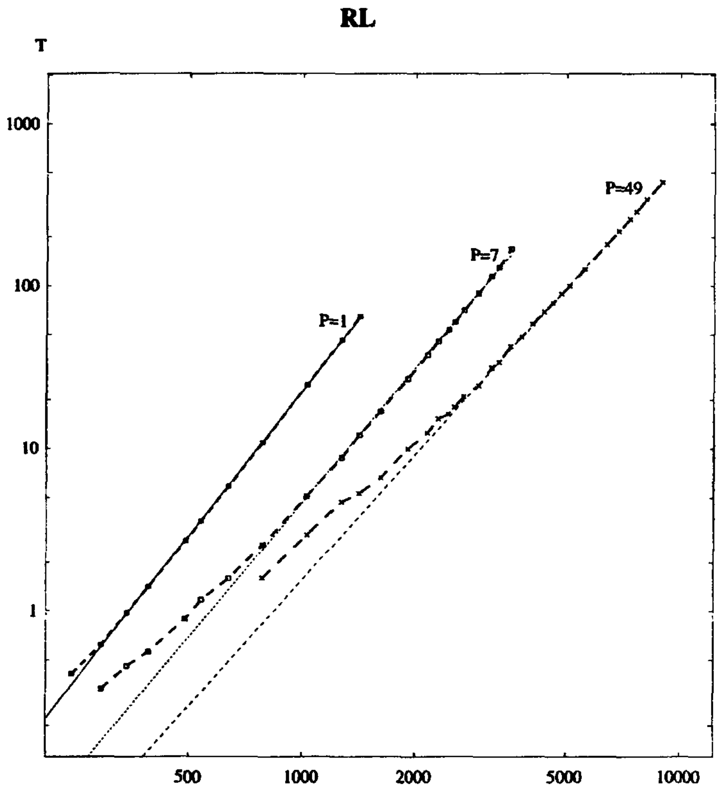


Figure 6. These two graphs show the time (in units of seconds) vs. the matrix order on 1, 7, and 49 processors with the methods of BL and BH. The points (connected by the dotted lines) are the collected data while the straight lines are the fitted results. The figures are a log-log plot.



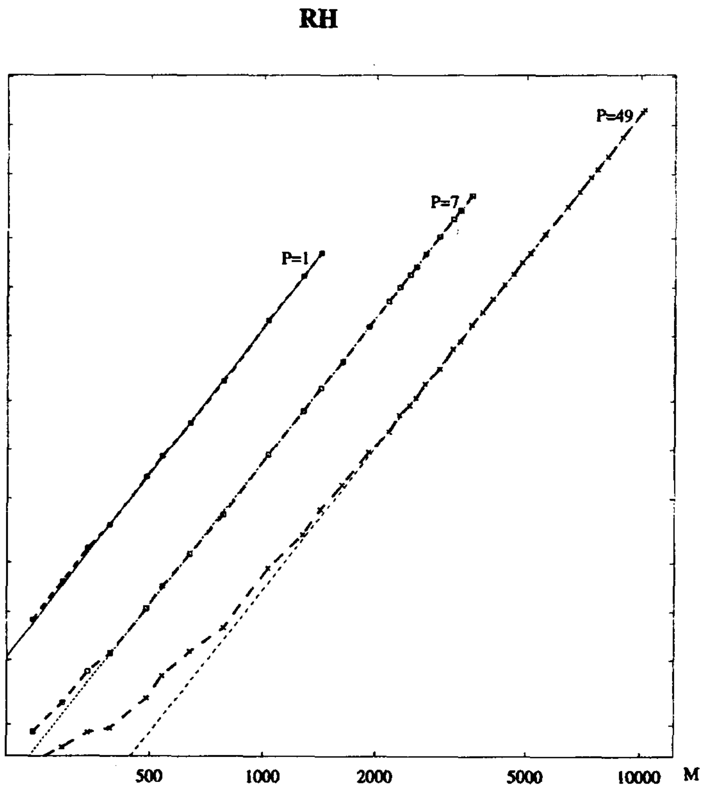


Figure 7. These two graphs show the time (in units of seconds) vs. the matrix order on 1, 7, and 49 processors with the methods of RL and RH. The points (connected by the dotted lines) are the collected data while the straight lines are the fitted results. The figures are a log-log plot.

Table 2. This table contains the six sets of coefficients α and β that appear in equation (5). Three sets are for the Ring method using 1, 7, and 49 processors and the other three are for our method using 1, 7, and 49 processors. The timing model is $T = \alpha M^{\beta}$, where α is in units of 10^{-9} seconds and β is dimensionless.

Processors	1		7		49	
Method\Coeff	α	β	α	β	α	β
SL	25.6	2.9815	9.80	2.8459	1.85	2.8146
SH	146	2.8342	45.3	2.7323	7.10	2.7311
BL	31.8	2.9502	13.1	2.8240	1.54	2.8644
BH	60.3	2.9914	33.4	2.8253	2.05	2.9368
RL	30.5	2.9565	25.6	2.7528	30.5	2.5685
RH	60.1	2.9921	12.4	2.9465	2.19	2.9330

Table 3. This table shows the time (in units of seconds) spent on MM for several matrix orders M using $P = 1, 7, 49$ processors. Using methods of SH and SL.

Methods	SL			SH		
M\P	1	7	49	1	7	49
128	0.10	0.22	0.30	0.14	0.13	0.17
256	0.44	0.27	0.31	0.97	0.26	0.19
400	1.50	0.45	0.34	3.52	0.66	0.25
480	2.50	0.58	0.32	5.81	1.01	0.31
512	3.04	0.70	0.38	6.87	1.16	0.32
640	5.94	1.11	0.44	12.97	2.07	0.47
800	11.48	1.91	0.57	24.93	3.86	0.75
1024	24.14	3.60	0.82	48.83	7.37	1.27
1152	34.27	4.96	0.99	68.37	10.24	1.70
1280	47.10	6.74	1.27	91.90	13.69	2.20
1408	62.94	8.77	1.55	124.25	17.85	2.81
1600		12.61	2.07		26.01	4.02
1664		14.13	2.27		28.97	4.46
1792		17.44	2.74		35.29	5.41
1920		21.32	3.26		42.71	6.49
2048		25.84	3.84		50.52	7.62
2176		30.86	4.66		60.22	9.19
2304		36.39	5.37		70.52	10.71
2432		42.74	6.26		82.12	12.42
2560		49.69	7.22		94.52	14.27
2816		66.06	9.39			18.53
3072			11.89			24.11
3328			14.94			29.90
3584			18.40			36.35
3840			22.40			43.91
4096			27.08			51.86
4352			32.27			61.71
4608			37.97			72.22
4864			44.48			83.98
4992			47.90			90.26
5120			51.57			96.63
5632			68.50			

(1) for small matrices, there exist large "start up" costs for distributing the submatrices and the timing does not follow any pattern, and

(2) when the matrices are sufficiently large (typical order: $M \geq 1000$ for 7 processors and $M \geq 2000$ for 49 processors), the timing does follow

$$T(M) = \alpha M^{\beta}. \tag{5}$$

This suggests that a fitting of our sufficiently many data points for large matrices to the above form should capture the timing behavior. The resulting lines in log-log plot are very straight and the coefficients (α and β) for these lines are tabulated in Table 2.

Table 4. This table shows the time (in units of seconds) spent on MM for several matrix orders M using $P = 1, 7, 49$ processors. Using methods of BH and BL.

Methods	BL			BH		
M\P	1	7	49	1	7	49
245	0.41	0.59	1.43	0.91	0.68	1.28
294	0.63	0.61	1.44	1.55	0.78	1.30
343	0.97	0.66	1.46	2.46	0.93	1.31
392	1.42	0.80	1.47	3.43	1.06	1.33
490	2.71	0.99	1.50	6.82	1.82	1.48
539	3.58	1.12	1.56	9.12	2.14	1.54
637	5.86	1.45	1.60	14.62	3.38	1.73
784	10.85	2.27	1.72	26.63	4.87	1.94
1029	24.59	4.35	2.10	62.99	11.37	2.94
1274	46.44	7.65	2.60	117.68	19.71	4.15
1421	65.06	10.47	3.04	162.99	27.64	5.31
1617		14.86	3.72		39.48	7.15
1911		23.84	5.05		61.02	10.28
2156		33.57	6.62		83.97	13.62
2303		40.80	7.74		105.02	16.70
2450		48.56	8.89		127.22	19.90
2548		54.60	9.79		137.02	21.37
2695		64.31	11.34		165.58	25.57
2940		82.36	13.94		207.83	31.68
3185		104.89	17.45		275.72	41.66
3332			19.48			45.21
3577			23.79			57.56
3822			28.54			68.63
4067			34.21			81.62
4361			41.84			101.52
4606			48.88			117.60
4851			56.76			135.96
5096			65.22			154.64
5586			84.73			206.09
6370			124.64			302.55
6860			155.04			370.50
7350			190.46			474.96
7644			213.69			526.13

The six sets of coefficients, three for Ring method and three for our method, are tabulated in Table 2. Using equation (5), we compute the timing ratios of our method to the Ring method at typical matrix orders on 1, 7, and 49 processors. The results are tabulated in Table 6.

Table 5. This table shows the time (in units of seconds) spent on MM for several matrix orders M using $P = 1, 7, 49$ processors. Using methods of RH and RL.

Methods	RL			RH		
M\P	1	7	49	1	7	49
490	2.69	0.89		6.81	1.05	0.29
539	3.58	1.17		9.12	1.43	0.40
637	5.86	1.58		14.61	2.26	0.57
784	10.85	2.54	1.59	26.64	3.98	0.80
1029	24.57	5.10	2.94	62.96	9.39	1.86
1274	46.43	8.76	4.71	117.56	17.34	2.97
1421	65.03	12.13	5.31	162.80	24.13	4.29
1617		16.97	6.61		34.98	6.07
1911		26.89	9.97		57.50	9.71
2156		37.48	12.51		82.31	12.96
2303		45.50	15.33		101.04	16.47
2450		53.71	16.32		120.35	18.83
2548		60.30	18.05		134.39	20.93
2695		70.84	20.86		160.05	25.62
2940		90.11	24.42		204.83	31.66
3185		114.26	31.41		263.59	41.77
3332		129.31	33.97		299.12	46.49
3577		167.85	42.41		370.97	58.82
3822			48.45			70.51
4067			58.33			85.33
4361			69.03			104.79
4606			78.63			121.38
4851			89.23			142.79
5096			99.84			161.92
5586			125.75			213.22
6370			179.29			314.35
6860			216.01			388.25
7350			258.20			477.89
7644			287.19			534.55
8134			342.56			642.08
8918			436.38			844.08
10045						1203.98
10192						1249.38

Table 6. This table shows the timing ratios of the three pairs of methods (SL, SH, BL, BH, RL, RH) tested in our study at typical matrix orders M on $P = 7, 49$ processors. The numbers in the row for method SL are the absolute times in units of seconds while the numbers in the remaining five rows are the time "in units of" the corresponding times taken by method SL.

Processors	$P = 7$			$P = 49$		
Methods\M	1000	2000	3000	4000	5000	6000
SL	$t_1 = 3.38$	$t_2 = 24.31$	$t_3 = 77.08$	$t_4 = 25.44$	$t_5 = 47.68$	$t_6 = 79.65$
SH	$2.11t_1$	$1.95t_2$	$1.87t_3$	$1.92t_4$	$1.88t_5$	$1.86t_6$
BL	$1.17t_1$	$1.15t_2$	$1.14t_3$	$1.26t_4$	$1.27t_5$	$1.28t_6$
BH	$2.96t_1$	$2.92t_2$	$2.89t_3$	$3.05t_4$	$3.14t_5$	$3.21t_6$
RL	$1.37t_1$	$1.29t_2$	$1.24t_3$	$2.14t_4$	$2.03t_5$	$1.94t_6$
RH	$2.53t_1$	$2.72t_2$	$2.83t_3$	$3.16t_4$	$3.25t_5$	$3.32t_6$

With the coefficients in Table 6 and the equation (5), we can compute the exact costs of multiplying certain matrix on $P = 7, 49$ processors by using any one of the six methods. The first half of the table shows the costs of multiplying matrices of orders $M = 1000, 2000, 3000$ on $P = 7$ while the next half the costs on orders $M = 4000, 5000, 6000$ on $P = 49$. In the first row (for method SL), we list the exact time in units of seconds, while in the remaining five rows list the times "in units of" the times in the corresponding first row. From this table, we notice (1) SL method is always 15–30% faster than BL, but about 30% (for $P = 7$) and about 100% (for $P = 49$) faster than RL.

5. CONCLUSIONS

Our tests suggest that a global parallelization of the S-method is always more efficient than the other popular methods: BMR and the Ring methods. It not only reduces the total number of floating-point operations, but also reduces the communication. The shortcomings for the S-method are:

(1) the number of processors is "quantized," which destroys the flexibility of using an arbitrary number of processors,

(2) the pattern of communication is quite random, which causes difficulty for implementation, and

(3) the S-method has some problems in numerical stability as recognized by other researchers [10].

This stability problem is not a serious problem and the gain in speed in using the S-method indeed justifies its value in applications where speed is important. We are in the process of reducing the problems in (1) and (2).

In the case of 49 processors, we have 32 submatrices. If we create three submatrix slots on each processor, each submatrix is repeatedly distributed for $3 \times 49/32 = 4.6$ times. We are developing a new scheme to distribute these submatrices based on their occurrence "frequency" in the 49 formulae for P_{ij} $\forall(i,j) \leq 7$. The more they appear, the more often we distribute them. This may yet reduce the cost of moving submatrices around.

In addition, we are applying the ideas presented in this paper to the Winograd method.

REFERENCES

1. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd edition, John Hopkins University Press, Baltimore, MD, (1989).
2. V. Pan, How can we speed up matrix multiplication?, *SIAM Reviews* **26**, 393–415 (1984).
3. V. Strassen, Gaussian elimination is not optimal, *Numer. Math.* **13**, 354–356 (1969).
4. D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, In *Proceeding of the Nineteenth Annual ACM Symposium on Theory of Computing*, pp. 1–6, (1987).
5. S. Winograd, Some remarks on fast multiplication of polynomials, In *Complexity of Sequential and Parallel Numerical Algorithms*, (Edited by J. Traub), pp. 181–196, Academic Press, (1973).
6. S.-T. Yau and Y.Y. Lu, Reducing the symmetric matrix eigenvalue problem to matrix multiplications, *SIAM Journal on Scientific Computing* **14**, 121–144 (1993).
7. K.A. Gallivan, R.J. Plemmons and A.H. Sameh, Parallel algorithms for dense linear algebra computations, *SIAM Rev.* **32**, 54–135 (1990).
8. U. Manber, *Introduction to Algorithms*, Addison-Wesley, Reading, MA, (1989).
9. D. Bailey, Extra high speed matrix multiplication on the CRAY-2, *SIAM J. Sci. Statist. Comput.* **9**, 603–607 (1988).
10. P. Bjørstad, F. Manne, T. Sørevik and M. Vajteršic, Efficient matrix multiplication on SIMD computers, *SIAM J. Anal. Appl.* **13**, 386–401 (1992).
11. D. Scott, Private communication, Supercomputer Systems Division of Intel Corporation, Beaverton, OR, (March 1994).
12. J. Choi, J.J. Dongarra and D.W. Walker, PUMMA: Parallel Universal Matrix Multiplication Algorithms on distributed memory concurrent computers, Technical Report TM-12252, Oak Ridge National Laboratory, (August 1993).
13. G.C. Fox, A.I. Hey and S. Otto, Matrix algorithms on the hypercube I: Matrix multiplication, *Parallel Computing* **4**, 17 (1987).

14. S. Huss-Lederman, A. Tsao, E.M. Jacobson and G. Zhang, Matrix multiplication on Intel Touchstone Delta, Technical Report: TR-93-101, SRC, (February 1994).
15. Kuck and Associates, *CLASSPACK Basic Math Library User's Guide*, Kuck & Associates, Champaign, IL, (December 1992).
16. G.C. Fox, M.A. Johnson, G. Lyzenga, S.W. Otto, J. Salmon and D. Walker, *Solving Problems on Concurrent Processors*, Vol. 1, General Techniques and Regular Problems, Prentice-Hall, Englewood Cliffs, NJ, (1988).
17. N.J. Higham, Exploiting fast matrix multiplication within the level 3 BLAS, *ACM Trans. Math Software* **16**, 112–115 (1990).