대형이산 행렬 시스템의 초대형병렬컴퓨터에서의 해법을 위한 병렬준비 행렬의 비교

마 상 백†

요 약

이 논문에서 우리는 CM-5와 같은 초대형병렬컴퓨터에서 대형 이산선형체제를 풀기 위한 준비행렬로써 두 가지를 소개한다. 대다수의 초대형병렬컴퓨터들은 프로세서간의 통신을 메세지패씽(message-passing)에 의존하는데 현재의 기술수준하에서는 이 통신속도가 실수계산속도에 비해 매우 느리므로 종래의 메모리공유컴퓨터에서와는 달리 데이터통신량을 최소화하는 알고리듬이 요구된다. 블록 SOR에 다중색채기법을 가미한 알고리듬이 그 한 예로써 우리는 이를 CM-5에서 구현한 결과 N=512x512 행렬에서 프로세서의 수가 16에서 512의 범위 하에서 50%의 효율을 실현하였다. 반면 종래의 효율적인 병렬 준비행렬로 알려진 ADI알고리듬은 방대한 량의 데이터통신 때문에 매우 열등한 결과를 보여준다.

Comparison of Parallel Preconditioners for Solving Large Sparse Linear Systems on a Massively Parallel Machine

Sang Back Ma †

ABSTRACT

In this paper we present two preconditioners for solving large sparse linear systems arising from elliptic partial differential equations on massively parallel machines, such as the CM-5. Most massively parallel machines do heavily rely on the message-passing for the interprocessor communications, but according to the current manufacturing standards the cost of communications is very high compared to that of floating point arithmetic computations. Due to this we need an algorithm which minimizes the amount of interprocessor communication on the massively parallel machines. We will show that Block SOR(SuccessiveOverRelaxation) method coupled with the multi-coloring technique is one of such preconditioner on the massively parallel machines, by conducting experiments on the CM-5. Also, we implemented the ADI (AlternatingDirectionImplicit) method on the CM-5, which has been conventionally one of the most powerful parallel preconditioner. Our experiment shows that Block SOR method coupled with the multi-coloring technique could yield a speedup with 50% efficiency with the range of number of processors from 16 to 512 for a matrix with dimension 512x512. On the other hand, the ADI method shows a very poor performance.

1. Introduction

Discretizations of elliptic PDE(Partial Differential Equation)s by FDM(Finite Difference Method) or FEM(Finite Element Method) in two and three dimensions lead to large sparse linear systems. Solving this linear system by conventional Gaussian elimination requires a prohibitive amount of memory and CPU time, especially on three dimensions. One alternative has been iterative methods, such as SOR[14], CG(ConjugateGradient)[1], or GMRES (Generalized Minimal RESidual) [10] method. Also, it is well known that preconditioning the given linear system $Ax = b$, by premultiplying the given linear system $Ax = b$, by a matrix M, $MAx = Mb$, is very effective in reducing the total amount

† 정 회 원:한양대학교 전자계산학과 전임강사
논문접수:1995년 3월 13일, 심사완료:1995년 7월 3일

of computations for a suitable choice of the preconditioning matrix M. There has been a lot of research on the preconditioner, and ILU(0)[7] is one of most popular preconditioner. However, most preconditioners are inherently serial, which leads to look for parallel preconditioners for various types of contemporary parallel computers.

ADI method is attracting new attentions due to its suitability to parallel computations. Its advantage is the linear solution of the tridiagonal system in H and V parts can be done in parallel. This method is very effective for a vector machine, such as the CRAY XMP, or CRAY-2, and its implementation is very straightforward. However, on message-passing machines such as the CM-5, we believe the ADI method suffers from the high amount of communications, and the methods such as Multi-Color Block SOR will have a superior performance than the ADI method.

2. ADI method

Finite difference or finite element discretizations of the following partial differential equation (PDE)

$$(K_1(x, y)u_x)_x-(K_2(x, y)u_y)_y+(d(x, y)u)_x+(e(x, y)u)_y+f(x, y)u=g(x, y)$$
$$\Omega=[0,1]\times[0,1] \quad (1)$$
$$u=0 \text{ on } \delta\Omega$$

with meshsize $h=1/(n+1)$ give rise to a linear system

$$Au=b \quad (2)$$

of order $N=n\times n$, where the the matrix A is a sparse matrix. The matrix A is nonsymmetric due to the presence of the terms of first order derivatives. In case of finite difference method(FDM) with standard central difference for the first-order derivatives, we could split, $A=H+V+\Sigma$, where H and V comes from the discretizations in x and y directions, respectively, and Σ comes from the term f in Eq. (1). For Eq. (1) we decompose A as $A=H_0+V_0+\Sigma$, where Σ come from the fu component, and H_0, V_0 are the contributions from the x, and y directional derivatives, respectively. With $H=H_0+(1/2)\Sigma$, and $V=V_0+(1/2)\Sigma$, **PR-ADI** (**P**eaceman-**R**achford **ADI**) method could be defined as

$$(H+\rho iI)ui_{+1/2}=-(V-\rho iI)ui+b \quad (3)$$
$$(V+\rho iI)ui_{+1}=-(H-\rho iI)ui_{+1/2}+b \quad (4)$$

where u_0 is an arbitrary initial vector approximation of the solution of Eq. (1), and $\{\rho i, i\geq 0\}$ are positive constants called acceleration parameters, which are chosen to speedup the convergence of this process. Each of Eq. (3) and (4) form n sets of linear system of order n where the n linear systems are completely *decoupled*. Furthermore, the matrices H and V could be made *tridiagonal* with proper reordering. For example, under natural ordering in x direction H is tridiagonal, and with natural ordering in y direction V could be made tridiagonal. This ensures a minimum degree of parallelism of n, which makes **PR-ADI** attractive in parallel computations. Also we note that Gaussian elimination method for the tridiagonal linear systems is very effective in terms of costs.

3. Multi-Color Block SOR method

3.1 Block SOR/SSOR methods

To define the Block SOR(Successive OverRelaxation) methods, we start by partitioning the NxN matrix A into the blocks as follows:

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,1} & \cdots & A_{2,n} \\ \vdots & \vdots & & \vdots \\ A_{n,1} & A_{n,2} & \cdots & A_{n,n} \end{bmatrix}$$

We assume that each diagonal block A^{ij}, $1 \leq i \leq n$, is a nonsingular matrix.

From the above partitioning we define the nxn block-partitioned matrices D, E, and F as

$$D^{ij} = \begin{cases} A^{ij}, \text{ if } i=j \\ 0, \textit{ otherwise} \end{cases}$$

Similarly, let E and F denote the lowertriangular and uppertriangular blocks, respectively, so that A=D+E+F. To solve Ax=b we define the Block SOR method as follows.

Algorithm Block SOR(BSOR)

1. Choose x_0
2. For i)0, ..., until Convergence Do

$(D-wE)x_{i+1}=((1-w)E+wF)x_i+wb$

Block SSOR(SymmetricSuccesiveOverRelaxation) method is defined as follows.

Algorithm Block SSOR(Symmetric Successive OverRelaxation)

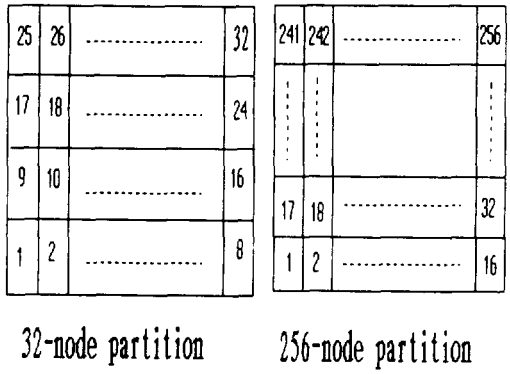
1. Choose x_0
2. For i)0, ..., until Convergence Do

$(D-wE)x_{i+\frac{1}{2}}=((1-w)E+wF)x_i+wb$

$(D-wE)x_{i+\frac{1}{2}}=((1-w)F+wE)x_{i+\frac{1}{2}}+wb$

w is called the relaxation parameter for acceleration of convergences, and usually chosen between 1 and 2. For diagonally dominant matrices, for example, the Block SOR/SSOR method is known to converge, if w is between 0 and 2[14].

In this paper we assume that the domain is a square, which is further divided into p rectangle-shaped blocks, where p is the number of the available processors. Further we assume in the most natural form as in (Fig. 1) that there is a one-to-one correspondence between the p blocks and p processors.



(Fig. 1) Mapping between a square domain and processors

3.2 Multi-color reordering

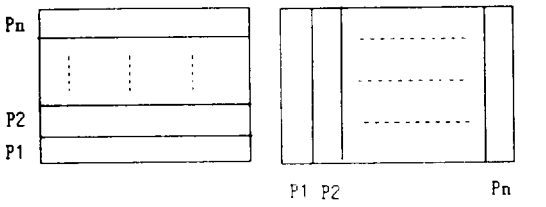
Given a mesh, multi-coloring consists of assigning a color to each point so that the couplings between two points of the same color are eliminated in the discretization matrix. For example, for the 5-point Laplacian on a square with two colors in a checkerboard fashion we can remove the coupling between any two points of the same color, so that the values at all points of one color can be updated simultaneously. Similarly, four colors are needed to color the grid points of the same color of the 9-point Laplacian. However, it has been known that the convergence rate for the reordered systems often deteriorates[4]. For the model problem SSOR and preconditioned-CG with the 2-color(Red/Black) ordering have a worse convergence rate than with the natural ordering, while SOR has the same rate if the optimal w is used. The following 〈Table 1〉 contains the

rates of convergence for SOR, SSOR, and ILU(0) preconditioned CG methods with natural and red/black ordering for the 5-point Laplacian matrix, when optimal w is used.

〈Table 1〉 Rate of convergence when reordering is used. h is the meshsize.

	SOR	SSOR	ILU-CG
Natural Ordering	$O(h)$	$O(h)$	$O(\sqrt{h})$
Red/Black Ordering	$O(h)$	$O(h^2)$	$O(h)$

For the nine-point Laplacian with properly selected w the convergence rate of SOR remains the same as with the natural ordering. O'Leary[8] has considered several other ordering schemes for the 9-point Laplacian and has shown that the convergence rate of SOR iteration is no worse than that of the natural ordering. This is the reason behind our choice of Block SOR preconditioner in combination with the multi-color reordering.



(Fig. 2) Block row/column mapping

4. Communication Costs

Let us consider the ADI method again.

$$(H+\rho_i I)u^{i+1/2} = -(V-\rho_i I)u^i + b \quad (5)$$

$$(V+\rho_i I)u^{i+1} = -(H+\rho_i I)u^{i+1/2} + b \quad (6)$$

Each of Eq. (5) and Eq. (6) has n independent linear systems, providing minimum parallelism of n. The implementation of ADI on machines with shared memory will be straightforward.

For example, on a single CPU of CRAY-2 as long as n exceeds 64, the vector length, satisfactory performance could be obtained. For message-passing machine the communication poses a problem. Let us consider the CM-5. Assume that the $N=n\times n$ grid is mapped into processors, $N_i, 1\le i\le p$, by Block Row/Column mapping as in (Fig. 2), ie., passing to the processor N_i, the n/p consecutive lines of n×n grid. For the MC-BSOR(Multi-Color Block SOR) method we used the form in the Multi-Color Block SOR method of Chapter 3 in [4]. Also, rather than iterating until convergence we iterate l times without checking the error criterion. We call this variant of MC-BSOR and ADI as MC-BSOR(l) and ADI(l). 〈Table 2〉 compares the communication related works of ADI(l) and BSOR(l). We note that the order of magnitude of the communication work of ADI is higher than that of the MC-BSOR(l).

〈Table 2〉 Comparisons of communication related work per one preconditioning step

	communication start-ups	data amount to be passed
ADI(l)	$4l$	O(N)
MC-BSOR(l)	l	$O(\sqrt{N})$

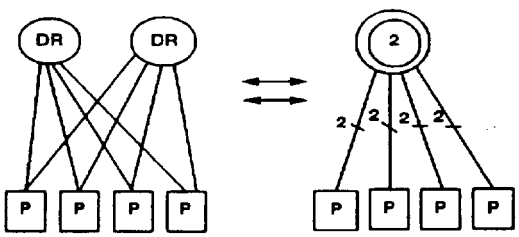
5. Overview of the CM-5

The Connection Machine(CM-5) from the Thinking Machines Corporation is a message-passing machine designed to support both SIMD and SPMD modes. The basic components of the CM-5 include hundreds or thousands of processing nodes, each with its own memory, one or more control processors, two global communication networks, high bandwidth I/O subsystems, and mass storage devices, eg., the data vault.

A CM-5 computer is divided into several

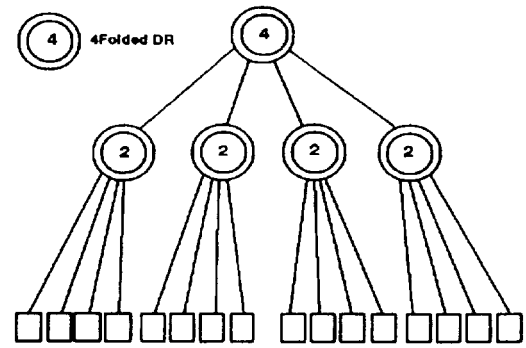
partitions. There is a separate processor, called control processor(CP), for each partition. A control processor is essentially like a standard high-performance workstation computer. It consists of a standard microprocessor, a SUN SPARC chip, separate memory, a network interface(NI) providing access to the communication networks, and other devices and interfaces. A control processor acting as a partition manager(PM) controls each partition and communicate with the rest of the system through the communication networks.

Every control processor and parallel processing node in the CM-5 is connected to two scalable interprocessor communication networks, the control network(CN) and the data network(DN). In general, the CN is used for operations that involve all the processors at once, for example, operations such as synchronizing and instruction broadcasting. The data network is used for bulk data transfers where data has a single source and destination. The CN also contains integer and logical arithmetic hardware for carrying out reduction operations, where every processor provides a value and all values are combined by the CN to produce a single result.



(Fig. 3) A 4-node hypertree and its simplified representation

The PNs are interconnected in a tree-like structure, as in (Fig. 1)-(Fig. 2). The parent nodes themselves are not PNs but Data Router(DR)s, and as we go up the level of the tree the communication channel capacity increases. This feature overcomes the usual communication bottleneck that often characterizes tree structures. In theory, the DN provides enough bandwidth for every network interface(NI) to sustain data transfer rates of 20Mbytes per second to any other NI within its group of four; 10 Mbytes to any other NI with its group of 16; and 5 Mbytes per second to any other in the system. Thus, the best to worst performance ratio is a factor of at most 4.



(Fig. 4) A 16-Node hypertree

At any time a processor may send a message to any other processor in the user task. This is done by first writing the destination processor number, and then the data to be sent, to the control registers in the NI. Once the DN has accepted the message, it assumes all responsibility for the delivery of the message to its destination. Moreover, the operation of the DN is independent of the PN's, which may carry out unrelated computations while the messages are in transit.

In the Single-Program-Multiple-Data (SPMD) mode on the CM-5, each PN holds an identical copy of the same program, called the node program, and executes its own copy concurrently.

But, unlike data parallel(SIMD) mode, different PNs can take different execution

branches. The host processor can execute a separate program independently. Data are exchanged through the DN. Interprocessor synchronization is automatically performed at points where processors are expected to communicate.

The CM-5 at Army High Performance Computing Research Center(AHPCRC), has 544 processing nodes, 5 control processors, and 120 Gbytes of mass storage. Each processing node has a 33 MHz SPARC processor from SUN Microsystems, 32 Mbytes of memory and 4 vector paths. The 544 PNs can be configured into two partitions of 512 and 32 PNs or three partitions of 32, 256, and 256 PNs. At the time when the experiments in this thesis were conducted vector units were not available.

6. Experiments

We consider the Elman problem on square grids in two dimensions. We discretized the problems using centered finite difference discretizations for FDM discretizations. The mesh sizes vary from test to test and are reported independently in this section. diagonals.

- Elman's problem [1]

$$-(bu^x)x-(cu^y)y+(du)x+(eu)y+fu= f(x,y)$$
$$\Omega=[0,1]\times[0,1] \quad (9)$$
$$u=0 \text{ on } \delta\ \Omega$$

where $b=exp(-xy)$, $c=exp(xy)$, $d=\beta(x+y)$, $e=\gamma(x+y)$, $f=1/(x+y)$, and g is such that exact solution $u=x\ exp(xy)\ sin(\pi x)\ sin(\pi y)$

The FGMRES(flexible GMRES) routine[11] allowing variable preconditioner at each iteration was used with $m=10$, $\varepsilon=10^{-6}$ for the *outer* iteration, where m is the dimension of Krylov subspace associated with the GMRES method.

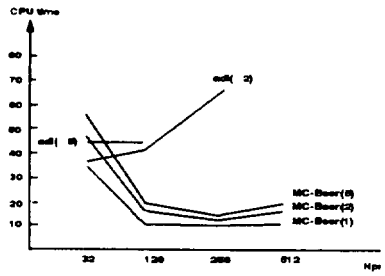
For our implementations on the CM-5 we have adopted the SPMD mode over data parallel mode. The reason is that the matrices coming from FDM or FEM have a special structure originating from the underlying physical problems. The communication pattern required by operations on such matrices, such as the matrix-vector multiplications, will take advantage of such a special structure.

The blocks were divided so that they are as close to squares as possible, to minimize the communication costs. All the communications were done with the **cmmd_swap** primitives, because swapping is expected to be more efficient than separate **send** and **receive**[15].

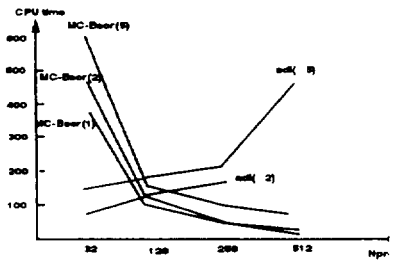
However, due to the **blocking** nature, ie, computation cannot go past the point of communications unless all the communications associated with that point are finished, deadlocks during the process of communication could happen. Since the original matrix A is structurally symmetric, this deadlock can avoided be by sorting the neighboring processors in an ascending order.

The number of colors needed for this matrix is 2, hence in this case Multi-Color BSOR reduces to the usual Red/Black SOR method. (Fig. 5) and (Fig. 6) compare Multi-Color BSOR(l) versus ADI(l), and $l=2,5$ with rectangular blocks of equal sizes. In(Fig. 5) the timing of the ADI(5) for number of processors, 256 and 512, were not reported, since it didn't finish within reasonable time. Timing was not measured in dedicated time, so it bears some inaccuracies due to the system load. $\omega=1.2$ was used through the experiments for Multi-Color BSOR(1) method, for simplicity. However, as we noted earlier

there is no simple way of finding the optimal in advance, and fortunately the performance of the SOR method is known relatively insensitive to the value of ρ[14].



(Fig. 5) Elman problem with FDM, $\gamma=50$, $\beta=1$, with FGMRES(10), N=256x256.

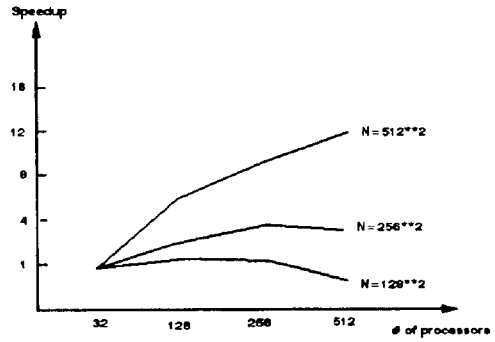


(Fig. 6) Elman problem with FDM, $\gamma=50$, $\beta=1$, with FGMRES(10) N= 512 x 512.

The number of available processors is 32, 128, 256, and 512. For 32 processors ADI(*l*) gives a better performance than Multi-Color Block SOR in terms of the total CPU time. However, as the number of processors increases the performance of ADI(*l*) worsens. In fact, for ADI(*l*) with a given problem, the CPU time increases with higher number of processors, which implies that speed-up is always smaller than 1.

Note that for a given problem the number of iterations increases with Multi-Color Block SOR, since the block size changes with number of processors, which also changes the convergence rate itself.

(Fig. 7) shows the speed-ups of FGMRES (10) with MC-BSOR(1) preconditioner on 32, 128, 256, and 512 PNs. The speed-up was measured relative to 32 nodes. For $N=512^2$ the speed-ups behave linearly for both problems.



(Fig. 7) Speed-up of FGMRES(10) /MCBSOR(1) for Elman problem, $\gamma=50$, $\beta=1$, $\omega=1.2$

These figures show linear speed-ups when the problem size is large enough relative to the number of PNs. This indicates that Multi-Color Block SOR(1) might have a good potential as a parallel preconditioner for the general case.

7. Conclusion

On our CM-5 experiments ADI(1) method suffers from the need for a high amount of communication. On a message-passing machine like the CM-5, Multi-Color Block SOR (*l*) method outperforms the ADI(*l*) method due to the high communication costs. For a matrix with the size of 512 x 512 we were able to get speed-ups of p/2(50% efficiency) relative to the case when p=32, with Multi-Color Block SOR method. On the other hand, the speed-up of ADI method is less than 1, for any number of processors.

On a message-passing machine we need to develop an algorithm which minimizes the amount of data movement between processors. Block SOR method combined with Multi-Color reordering technique were able to produce a speed-up curve of 50% efficiency for

our test problem from a partial differential equation.

References

[1] H. Elman, "Iterative methods for large, sparse, nonsymmetric systems of linear equations," Ph. D Thesis, Yale University, 1982.

[2] W. Ferng, K. Wu, S., Petiton, Y. Saad, "Basic sparse matrix computaions on massively parallel computers," UMSI 92/084, AHPCRC, University of Minnesota, 1982.

[3] W. J. Layton and P. J. Rabier, "Peaceman Rachford procedure and domain decomposition for finite element problems," Technical Report, University of Pittsburgh, Pittsburgh, PA, 1991.

[4] C. -C. Jay Kuo and Tony Chan, "Two-color Fourier analysis of iterative algorithms for elliptic problems with red/black ordering", SIAM J. Sci. Stat, Vol. 11, No. 4, pp. 767-793, 1990.

[5] S. Ma, "Parallel block preconditioned Krylov subspace methods for partial differential equations," Ph. D Thesis, University of Minnesota, Minneapolis, Aug, 1993.

[6] S. Ma and Y. Saad, "Block-ADI preconditioners for solving sparse nonsymetric linear systems of equations," Numerical Linear Algebra, Edited by L. Reichel, a. Ruttan and R. Varga, Walter de Gruyter, 1993.

[7] J. Meijerink and H. Van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix", Math. Comp., Vol. 31, pp. 148-162, 1977.

[8] D. O'Leary, "Ordering schemes for parallel processing of certain mesh problems", SIAM J. Sci. Stat Vol. 5, pp. 620-632, 1984.

[9] D. Peaceman and H. Rachford, "The numerical solution of elliptic and parabolic differential equations," Journal of SIAM., Vol. 3, pp. 28-41, 1955.

[10] Y. Saad and M. Schultz, "GMRES : A Generalized minimal residual algorithm for solving nonsymmetric linear systems," SIAM J. Sci. Stat, Vol. 7, July, 1986.

[11] Y. Saad, "Highly parallel preconditioners for general sparse matrices," Technical Report UMSI 92/45, University of Minnesota, Army High Performance Computing Research Center, Minneapolis, Minnesota, 1992.

[12] Y. Saad, "Krylov subspace methods in distributed computing environments," Army High Performance Computing Research Center, 92-126, 1992.

[13] Technical summary : CM-5, Thinking Machine Corporation, Cambridge, Massachusetts

[14] R. Varga, Matrix Iterative Analysis, Prentice-Hall, New York, 1962

[15] User's Guide of CM-Fortran, Thinking Machine Corporation, Cambridge, Massachusetts

[16] D. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971

마 상 백

1978년 서울대학교 계산통계학과 졸업

1978년~83년 국방과학연구소 연구원

1983년~87년 미국 미네소타 주립대학 수학 석사

1987년~93년 미국 미네소타 주립대학 전자계산학 박사

1994년~현재 한양대학교 공과대학 전자계산학과 전임강사

관심분야 : 수치해석, 병렬알고리즘