J. KSIAM **Vol.5**, No.1, 85-100, 2001

A Robust Preconditioner on the CRAY-T3E for Large Nonsymmetric Sparse Linear Systems

Sangback Ma and Jaeyoung Cho

Abstract

In this paper we propose a block-type parallel preconditioner for solving large sparse nonsymmetric linear systems, which we expect to be scalable. It is Multi-Color Block SOR preconditioner, combined with direct sparse matrix solver. For the Laplacian matrix the SOR method is known to have a nondeteriorating rate of convergence when used with Multi-Color ordering. Since most of the time is spent on the diagonal inversion, which is done on each processor, we expect it to be a good scalable preconditioner. Finally, due to the blocking effect, it will be effective for ill-conditioned problems. We compared it with four other preconditioners, which are ILU(0)-wavefront ordering, ILU(0)-Multi-Color ordering, SPAI(SParse Approximate Inverse), and SSOR preconditioner. Experiments were conducted for the Finite Difference discretizations of two problems with various meshsizes varying up to 1024 x 1024 , and for an ill-conditioned matrix from the shell problem from the Harwell-Boeing collection. CRAY-T3E with 128 nodes was used. MPI library was used for interprocess communications. The results show that Multi-Color Block SOR and ILU(0) with Multi-Color ordering give the best performances for the finite difference matrices and for the shell problem only the Multi-Color Block SOR converges.

1 Introduction

Iterative solutions of large sparse linear systems require the use of preconditioning techniques in order to converge in a reasonable number of iterations. Reordering the equations through *multi-coloring* provides a parallelism of order N, where N is the dimension of the given matrix. For example, if the matrix has property A, as is the case for the standard 5-point matrices obtained from centered Finite Difference (FD) discretizations of elliptic Partial Differential Equations (PDE's), there is a partition of the grid-points in two disjoint subsets such that the unknowns of any one subset are only related to unknowns of the other subset. This enables to produce a reordered matrix having a block-tridiagonal matrix, where the diagonal blocks are diagonal matrices. There are several different ways of exploiting this structure. For example, the unknowns associated with one of the subsets can be easily eliminated, and the resulting reduced system is often well-conditioned. This 'two-coloring' often referred to as a red-black

or checkerboard ordering, can be generalized to arbitrary sparse matrices by using multi-coloring. But the drawback of this approach is that it often suffers from the deterioration of the rate of convergence with certain iterative methods, such as in preconditioned CG(Conjugate Gradient) method.

Wavefront or *level scheduling*[11] technique is still another way to achieve a parallelism while preserving the initial ordering. It does not suffer from the deterioration of the rate of convergence as with multi-coloring, but the maximum parallelism is determined by the lengths of the wavefront, which is often nonuniform.

In the SPAI approach a sparse approximate inverse is computed explicitly, and then applied as a preconditioner to an iterative method. The computation of the preconditioner is inherently parallel, and its application only requires a matrix-vector product. The sparsity pattern of the approximate inverse can be fixed in advance, or expanded for more accuracy. The SPAI computes the entries, M, so that it minimizes $\|AM - I\|$ under suitable norm, often the Frobenius norm. It decomposes into independent least squares problems, which can be solved in parallel. There are many variations in the SPAI approach depending on the way the residual norm is minimized and the choice of sparsity pattern. In this paper we adopted the approach by Grote and Huckcle[4].

The Multi-Color Block SOR method combines the Multi-Coloring with the Block SOR method. It is known that for the 5-point Laplacian the SOR method has the same rate of convergence even with the Multi-Coloring, while the ILU(0) preconditioned CG method has a low rate of convergence with the Multi-Coloring. Hence, we expect the Multi-Color Block SOR method to be a good choice.

The CRAY-T3E computer in ETRI, Korea is a massively parallel message-passing machine with the 136 individual processing node(PE)s interconnected in a 3D-Torus structure. Each PE, a DEC Alpha EV5.6 chip, is capable of delivering up tp 900 Megaflops,amounting to 115 GigaFlops in total. Each PE has 128 MBs of core memory.

2 Multi-coloring and Wavefront reordering

2.1 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 the 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 9-point Laplacian. However, it has been known that the convergence rate for the reordered systems often deteriorates. For the model problem SSOR and preconditioned-CG with the Red/Black ordering have a worse convergence rate than with the natural ordering, while SOR has the same rate if optimal ω is used. The table 1 contains the rates of convergence for SSOR with optimal ω, and ILU(0) preconditioned CG methods with natural and

red/black ordering for the 5-point Laplacian matrix.[5]

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)

The Red/Black ordering can be extended to Multi-Color ordering schemes. For the nine-point Laplacian with properly selected ω the convergence rate of SOR remains the same as with the natural ordering. O'Leary[7] 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 with the natural ordering.

However, even though the performance is not as sensitive to ω as it is to ρ in ADI, it is preferable to have an optimal ω for the best performance as a preconditioner. For model problems with the 5-point, Laplacian the optimal ω both with the natural ordering and red/black ordering has been determined, but the determination of such ω is very difficult, in general. Also, when the points of one color is being updated, the other points are idle. This could cause CPU idle in parallel execution, depending on the actual mapping between the grid points and the processors.

Regarding the blocked version of SOR, blocking in general increases the convergence rate while the cost of one solution per iteration increases. For the model problem with the 5-point Laplacian the convergence rate of the line-SOR method is $2\sqrt{2}\pi h$, $\sqrt{2}$ times that of point SOR method. If the square was divided into L subsquares, each with size of $q \times q, q = n/\sqrt{L}$, the spectral radius of Block Jacobi has been found to be[2]

$$Sp(M_J) \approx 1 - q\pi^2 h^2/2, \tag{1}$$

where M_J is the Block Jacobi iteration matrix. ¿From the relation about optimal ω and the spectral radius of the Block SOR iteration matrix [15] the optimal $\omega^* \approx 2/(1+\sqrt{q}\pi h)$, and hence the rate of convergence is $2\sqrt{q}\pi h$, $\sqrt{q}$ times that of point SOR. On the other hand since the diagonal blocks are no longer diagonal matrices, inverting them involves some extra costs.

As preconditioners SOR and Block SOR methods are expected to perform well as good preconditioners, although analytic explanation is not available. We could combine Block SOR with Multi-coloring for parallel preconditioners.

Let the domain be divided into L blocks. Suppose that we apply a multi-coloring technique, such as a greedy algorithm described in [14],to these blocks so that a block of one color has no coupling with a block of the same color. Let D_j be the coupling within the block j, and the $E_{j,k}, k = 1, q, k \neq$ color(j) the coupling between the j-th block and the other blocks of color k, where color(j) is the color of the block j.

Then, we can describe the Multi-Color Block SOR as follows.

ALGORITHM **2.1 Multi-Color Block SOR**

Let q be the total number of colors, and color(i), i=1, L, be the array of the color for each block.

1. *Choose* u_0, *and* $\omega > 0$.
2. *For* $i > 0$ *Until Convergence Do*
3. *For kolor = 1, q Do*
4. *For j = 1, L Do*
5. *if(color(j) == kolor) then*
6. $(u_{i+1}^{GS})_j = {D_j}^{-1}(b - \sum_{k=1}^{k=q-1} E_{j,k}u_i)$.
7. $u_{i+1} = u_i + \omega * (u_{i+1}^{GS} - u_i)$.
8. *endif*
9. *Endfor*
10. *Endfor*
11. *Endfor*

The u_i^{GS} is the i-th update of Block Gauss-Seidel iteration. If $\omega = 1$ it is equivalent to Block Gauss-Seidel method. Note that the innermost loop in line six and seven can be executed in parallel.

2.2 Wavefront-ordering(Level scheduling)

Rather than pursuing the parallelisms through reordering, the wavefront technique exploits the structure of the given matrix. If the matrix comes from the discretizations of PDEs such as by FDM or FEM, the value of a certain node is usually dependent on only the values of its neighbors. Hence, once the values of its neighbors are known that node can be updated.

For example, let us assume that we have an ILU(0) factorization for the 5-point Laplacian for the Poisson equation, and for a preconditioner we need to solve

$$Lz = y,$$

and

$$Ux = z$$

.

Wavefront technique(or Level scheduling) is a process of finding new ordering of the nodes so that the new matrix would look like as in Fig. (1), where the L_i's are diagonal blocks. This technique would work equally well for three dimensional problems as well as two dimensional problems. For references, see [11]

Figure 1: Block partitioning for L

3 SPAI Preconditioner

We look for M such that $\|AM - I\|_F$ is minimized where $\|\|_F$ stands for the Frobenius norm. Then since

$$\|AM - I\|_F{}^2 \;=\; \sum_{k=1}^{n} \|(AM - I)e_k\|^2 \tag{2}$$

finding such M separates into n independent least squares problems

$$\min_{m_k} \|Am_k \;-\; e_k\|_2 \;, k = 1, \ldots, n, \tag{3}$$

where e_k has 1 in the k-th position and 0 elsewhere. If M is sparse, Eq. (3) becomes n small least squares problems, which can be solved quickly.

The following is a description of the SPAI method as in [4].

Let τ be a given parameter controlling how much the approximate inverse is to be close to the actual inverse.

(a) Choose an initial sparsity pattern J, where $m_k(j) \neq 0,\ j \in J$.

(b) Compute the row indices I such that $A(i, J)$ is not identically zero, $i \in I$. Let $\hat{A} \;=\; A(I, J),\ \hat{e_k} = e_k(I)$. Find the QR decomposition of $\hat{A}$,

$$\hat{A} \;=\; Q \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{4}$$

Then, solve

$$\min_{\hat{m_k}} \|\hat{A}\hat{m}_k \;-\; \hat{e_k}\|_2 \tag{5}$$

and the residual $r = A(.,J)\hat{m}_k - e_k$.

While $\|r\|_2 > \tau$

(c) Set K equal to $\{l|r(l) \neq 0\}$.

(d) Set $\tilde{J}$ equal to the set of all new column indices of A that appear in all K rows but not in J.

(e)For each $j \in \tilde{J}$ find the direction u_j such that $\|r + u_j Ae_j\|_2$ is minimized.

(f)For each $j \in \tilde{J}$ compute the 2-norm of the new residual $r + u_j Ae_j$ with the above u_j. Delete from $\tilde{J}$ all but the indices leading to significant reduction in 2-norm.

(g) Determine the new indices $\tilde{I}$ and update the QR decomposition accordingly. Then, solve the new least squares problem, compute the new residual $r = Am_k - e_k$ and set $I = I \cup, \tilde{I},\ J = J \cup \tilde{j}$.

For further details see [4].

4 ILU(0) factorization

Meijerink and Van. der Vorst[6] introduced a so-called Incomplete LU(ILU) preconditioner for symmetric matrices. The following is a modification of the original ILU for nonsymmetric matrices, as described in [3]. Let $A = LU + N$, where $L_{i,j} = U_{i,j} = 0$ if $A_{i,j} = 0$ and $N_{i,j} = 0$ if $A_{i,j} \neq 0$. Let NZ(A), the *nonzero pattern* of A, denote the set of pairs of (i,j) for which $A_{i,j}$, the (i,j) entry of A, is nonzero.

ALGORITHM **4.1 ILU Factorization**

1. *For i = 1, . . . , Until N Do*
2. *For j = 1, . . . , Until N Do*
3. *If((i,j) belongs to NZ(A)) then*
4. $s_{i,j} = A_{i,j} - \sum_{t=1}^{min(i,j)-1} L_{i,t}U_{t,j}$.
5. *if*$(i \geq j)$ *then* $L_{i,j} = s_{i,j}$.
6. *if*$(i < j)$ *then* $U_{i,j} = s_{i,j}/L_{i,i}$.
7. *Endif*
8. *Endfor*
9. *Endfor*

4.1 Point-SSOR algorithm

ALGORITHM **4.2** *Point-SSOR Let* $A = D - E - F$, *where* D *is the diagonal part,* $-E$, *is the lowertriangular part and* $-F$ *the uppertriangular part.*

1. *Choose* x_0.

2. For i=0, ... Do

$$\begin{aligned}(D-\omega E)x_{i+\frac{1}{2}} &= ((1\ -\ \omega)D\ +\ \omega F)x_i+\omega b \\ (D-\omega F)x_{i+1} &= ((1\ -\ \omega)D\ +\ \omega E)x_{i+\frac{1}{2}}+\omega ba\end{aligned} \tag{6}$$

Endfor

5 Experiments

5.1 Test problems

- **Problem 1** Elman's problem [3]

$$\begin{aligned}-(bu_x)_x-(cu_y)_y+(du)_x+du_x+(eu)_y+eu_y+fu &= g \\ \Omega &= (0,1)\times(0,1) \\ u &= 0\ \ on\ \ \delta\Omega\end{aligned} \tag{7}$$

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

- **Problem 2** Convection-diffusion equation

$$\begin{aligned}-\epsilon\triangle u+\cos\alpha\, u_x+\sin\alpha\, u_y &= f, \\ \Omega &= (0,1)\times(0,1) \\ u &= 0\ \ on\ \ \delta\Omega\end{aligned} \tag{8}$$

$$\tag{9}$$

- **Problem 3** Cylindrical shell problem from Harwell-Boeing Collection [1] The 's3dkq4m2.dat' from the CYLSHELL set.

We have set $\epsilon = 0.0001$, and $alpha = 15$ degree. Upwind scheme was adopted for the convection term. 5-point approximation was used for 2-nd order derivatives in Problem 1, and 9-point Laplacian for Problem 2. For problem 3 N = 90449 and the total number of nonzeros is equal to 4820891. This matrix is very ill-conditioned with an estimated condition number of 1.35^{11}.

Figure 2: Mapping between a square domain and processors

5.2 Domain mapping onto the processors

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 that there is a one-to-one correspondence between the p blocks and p processors. See Fig. (2).

5.3 Results

BICGSTAB[17] was used as the outer iterative method. As for the manipulation and storing of the matrices the CSR(Compress Sparse Row) format[13] was adopted, and the MSR(Modified Sparse Row) format for the ILU(0) factorization and forward/backward solves. Since CSR format was used, our experiments realistically simulates the unstructured problems even though we used the unit square as our domain. We used the wavefront-ordering for the point-SSOR method and the parameter ω was set to 1.2. The τ parameter for the SPAI, which controls how much the approximate inverse is going to be close the actual inverse was set to 0.4.

We set $\gamma = 50,\ \beta = 1,\ \epsilon = 0.1,\ \alpha = 15$, so that the resulting matrices for Problem 1 and 2 are nonsymmetric. For the multi-coloring we used a simple heuristic based on a greedy algorithm as in [14]. Using this heuristic the number of colors required for Problem 1 and 2 are 2 and 4, respectively.

For the Multi-Color Block SOR method we used the MA48 package to invert the diagonal block. For the partitioning of the 's3dkq4m2' matrix of the shell problem we have used the Metis 4.0 library by V. Kumar.

MPI(Message Passing Interface) library was used for the communications. This enables the codes to be run independent of the machines.

The CPU time and the number of iterations are shown in the Tables. 2 - 10. 'X' stands for the case where the memory was insufficient for the problem size. As we compare the number of iterations in wavefront order and that in multi-coloring order, we see little difference, unlike in the symmetric case, such as in the ILU(0)-preconditioned CG method. Hence, we are led to believe that for problems tested the multi-coloring order outperforms the wavefront order, which is verified in the tables. The SPAI preconditioner is hardly competitive. In all cases ILU(0) with the multi-coloring order outperforms the other preconditioners, except the Multi-Color Block SOR. Since the cost of inverting and back-solving by MA48 is approximately proportional to $(N/p)^2$, for a given N increasing p reduces the cost quadratically. Hence, we observe that with for a given N there ia a limit on p such that above this limit the Multi-Color Block SOR preconditioner outperforms the ILU(0) with the Multi-Color ordering. The CPU time does not include the proprocessing costs of generating the preconditioner matrices. For the 's3dkq4m2' matrix from the shell problem only the Multi-Color Block SOR converges. That matrix is veri ill-conditioned, with the condition number of 10^{-11}.

Table 2: Problem 1, with FDM, N=128x128

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	2.41/6	0.71/6	0.33/6	0.18/6	0.25/8
SSOR-WAVEFRONT	1.42/62	1.04/61	1.07/61	1.36/63	1.86/61
ILU(0)-WAVEFRONT	0.99/50	0.80/53	0.90/54	1.07/50	1.59/52
ILU(0)-MULTICOL	0.95/50	0.70/53	0.66/54	0.69/51	0.97/53
SPAI, $\tau = 0.2$	2.82/64	1.69/62	1.23/64	1.32/62	1.22/62
SPAI, $\tau = 0.4$	4.28/144	2.41/142	1.78/137	1.50/135	4.14/153

Table 3: Problem 1 with FDM, N=256x256

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	18.3/7	6.14/7	2.48/9	0.97/9	0.64/11
SSOR-WAVEFRONT	11.22/126	6.46/122	4.92/132	4.04/119	4.78/121
ILU(0)-WAVEFRONT	7.04/107	4.80/107	3.53/107	3.21/104	4.13/105
ILU(0)-MULTICOL	7.90/107	4.47/108	2.86/107	2.11/101	2.33/105
SPAI, $\tau = 0.2$	21.96/137	12.48/171	6.90/136	4.80/137	4.35/141
SPAI, $\tau = 0.4$	29.16/276	16.71/296	9.10/281	5.30/264	8.22/271

Table 4: Problem 1 with FDM, N=512x512

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	X	X	21.0/12	7.6/12	3.6/15
SSOR-WAVEFRONT	X	43.01/258	24.75/256	17.63/257	15.26/256
ILU(0)-WAVEFRONT	X	32.05/234	18.91/234	13.12/224	12.15/226
ILU(0)-MULTICOL	X	30.42/230	17.20/238	9.89/227	7.70/223
SPAI, $\tau = 0.2$	214.24/330	95.58/322	53.38/312	30.02/302	19.25/318
SPAI, $\tau = 0.4$	244.74/571	117.24/538	66.16/572	52.89/562	23.70/553

Table 5: Problem 1 with FDM, N=1024x1024

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	X	X	X	68.4/17	25.2/18
SSOR-WAVEFRONT	X	X	X	106.48/534	73.81/563
ILU(0)-WAVEFRONT	X	X	X	80.16/482	53.69/446
ILU(0)-MULTICOL	X	X	X	67.09/467	41.02/492
SPAI, $\tau = 0.2$	X	X	X	298.56/640	132.05/650
SPAI, $\tau = 0.4$	X	X	X	316.43/1373	159.20/1254

Table 6: Problem 2 with FDM, N=128x128

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	4.46/3	1.31/3	0.55/3	0.4/5	0.34/5
SSOR-WAVEFRONT	1.93/65	1.54/68	1.68/66	1.96/66	3.09/66
ILU(0)-WAVEFRONT	0.93/37	0.81/41	0.97/41	1.14/41	1.94/43
ILU(0)-MULTICOL	0.95/38	0.78/42	0.78/41	0.93/44	1.41/46
SPAI, $\tau = 0.2$	3.57/68	2.04/68	1,41/68	1.55/68	2.22/68
SPAI, $\tau = 0.4$	5.40/194	3.51/194	2.00/201	2.29/211	3.62/204

Table 7: Problem 2 with FDM, N=256x256

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	39.5/3	12.4/3	4.37/3	1.55/4	0.73/4
SSOR-WAVEFRONT	14.63/127	7.86/123	6.26/126	5.95/128	7.54/129
ILU(0)-WAVEFRONT	7.08/73	4.31/80	3.35/76	3.49/83	4.51/82
ILU(0)-MULTICOL	7.11/73	4.12/81	2.87/79	2.39/78	8.22/155
SPAI,$\tau = 0.2$	28.08/141	14.82/141	8.24/141	5.13/141	4.91/141
SPAI,$\tau = 0.4$	41.86/388	24.09/426	11.28/376	7.08/389	7.87/411

Table 8: Problem 2 with FDM, N=512x512

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	X	X	39.0/3	13.2/4	4.8/4
SSOR-WAVEFRONT	X	52.87/237	31.42/232	22.98/239	22.02/244
ILU(0)-WAVEFRONT	X	27.00/148	16.93/148	12.41/151	12.19/150
ILU(0)-MULTICOL	X	25.95/147	14.25/141	9.68/150	8.21/155
SPAi,$\tau = 0.2$	226.10/282	115.00/285	59.02/281	32.58/285	20.23/279
SPAi,$\tau = 0.4$	345.17/815	172.96/816	86.13/784	44.00/749	30.28/779

Table 9: Problem 2 with FDM, N=1024x1024

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	X	X	X	SLOW	42.4/5
SSOR-WAVEFRONT	X	X	X	128.16/471	88.64/459
ILU(0)-WAVEFRONT	X	X	X	61.40/270	46.19/279
ILU(0)-MULTICOL	X	X	X	55.69/285	34.92/289
SPAI, $\tau = 0.2$	X	X	X	341.54/535	159.33/535
SPAI, $\tau = 0.4$	X	X	X	442.01/2000	202.47/1724

Table 10: 's3dkq4m2' from the Harwell-Boeing Collection

	p = 4	p = 8	p = 16	p = 32	p = 64
	Cpu time/Iterations				
MC-BSOR(2)	X	X	X	1286.0/524	457.0/497
SSOR-WAVEFRONT	X	X	X	SL	SL
ILU(0)-WAVEFRONT	X	X	X	SL	SL
ILU(0)-MULTICOL	X	X	X	SL	SL
SPAI, $\tau = 0.2$	X	X	X	SL	SL
SPAI, $\tau = 0.4$	X	X	X	SL	SL

Table 11: Problem 1 with FDM, N=256x256

	p = 4	p = 8	p = 16	p = 32	p = 64
	Megaflops				
ILU(0)-WAVEFRONT	66	109	149	159	125
ILU(0)-MULTICOL	66	119	184	234	221

Table 12: Problem 1 with FDM, N=512x512

	p = 4	p = 8	p = 16	p = 32	p = 64
	Megaflops				
ILU(0)-WAVEFRONT	X	144	243	336	366
ILU(0)-MULTICOL	X	149	272	451	595

Table 13: Problem 1 with FDM, N=1024x1024

	p = 4	p = 8	p = 16	p = 32	p = 64
	Megaflops				
ILU(0)-WAVEFRONT	X	X	X	473	683
ILU(0)-MULTICOL	X	X	X	547	943

Table 14: Problem 2 with FDM, N=256x256

	p = 4	p = 8	p = 16	p = 32	p = 64
	Megaflops				
ILU(0)-WAVEFRONT	78	140	171	179	137
ILU(0)-MULTICOL	77	148	207	246	586

Table 15: Problem 2 with FDM, N=512x512

	p = 4	p = 8	p = 16	p = 32	p = 64
	Megaflops				
ILU(0)-WAVEFRONT	X	165	263	366	371
ILU(0)-MULTICOL	X	171	298	467	568

Table 16: Problem 2 with FDM, N=1024x1024

	p = 4	p = 8	p = 16	p = 32	p = 64
	Megaflops				
ILU(0)-WAVEFRONT	X	X	X	530	728
ILU(0)-MULTICOL	X	X	X	617	997

Figure 3: Speedup of problem 1, N= 512x512

6 Summary

- Unlike in the symmetric case the convergence rate of the BICGSTAB method deos not deteriorate at all, compared with the wavefront-order. Hence, due to the parallelism of order(N) coming from the multi-coloring,for problems tested ILU(0) with the multicolor ordering outperforms the other preconditioners considered in this paper. The Multi-Color Block SOR gives the worst performances.
- Seeing from the speedup curves, the communication overheads in the CRAY-T3E turns out to be very high for the irregularly structured sparse matrices in the CSR format. One of the reasons is that we are sorting the adjacent processor list array to avoid deadlocks. For example, in matrix-vector product this causes the higher numbered processors to wait until all of the lower numbered processors are done.
- It is an interesting fact on its own that the convergence rate of the BICGSTAB method in the multi-coloring order does not deteriorate, compared to that of the natural ordering. In future we might need more research to explain this phenomenon.

Acknowledgements. The work was supported by Korea STEPI research fund, 97-NF-03-01-A-03. The author would like to acknowledge the numerous advices of Prof. Youcef Saad, and also the Korea ETRI, which provided the computer facilities and an excellent research environment to conduct this research.

References

[1] M. benzi, R. Kouhia, and M. Tuma, "An Assessment of Some Preconditioning Techniques in Shell Problems", Communications in Numerical Methods in Engineering, Vol. 14, 1998, pp. 897-906

[2] D. Boley, B. Buzbee, and S. Parter, "On block relaxation techniques", MRC Technical Summary Report # 1860, Mathematics Research Center, University of Wisconsin, 1978.

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

[4] M. Grote and T. Huckle, "Parallel preconditioning with sparse approximate inverses", J. Sci. Computing, Vol. 18, 1997, pp. 838-853

[5] 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, 1990, pp. 767-793.

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

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

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

[9] Y. Saad, "Krylov subspace methods for solving large unsymmetric linear systems", *Mathematics of Computation*, Vol. 37, July 1981

[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, "Krylov subspace methods on supercomputers", *SIAM J. Sci. Stat*, Vol. 10, 1989, pp. 1200-1232.

[12] Y. Saad, "ILUT: A dual threshold incomplete LU factorization", Numer. Linear Algebra Appl.,Vol. 1, 1992, pp.387-402.

[13] Y. Saad, "SPARSKIT: A basic tool kit for sparse computations", in Iterative Methods for Sparse Linear Systems, PWS Publishing Company, Boston, 1996

[14] Y. Saad, "Highly parallel preconditioners for general sparse matrices", in Recent Advances in Iterative Methods, IMA Volumes in Mathematics and its Applications, Vol. 60, G. Golub, M. Luskin and A. Greenbaum, eds, Springer-Verlag, Berlin, 1994, pp. 165-199.

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

[16] H. Van Der Vorst, "High performance preconditioning", *SIAM J. Sci. Stat*, Vol. 10, 1989, pp. 1174-1185.

[17] H. Van der Vorst, "BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems", SIAM. J. Sci. Statist. Comput., Vol. 13, 1992, pp. 631-644.

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

Department of Computer science
Hanyang University, Sa-1 Dong 1271
Ansan, Kyungki-Do, S. Korea, 425-791
email :sangback@cse.hanyang.ac.kr