대형비대칭 이산행렬의 CRAY-T3E에서의 해법을 위한 확장가능한 병렬준비행렬

마 상 백†

요 약

본 논문에서는 대형비대칭 이산행렬의 해법을 위한 확장가능한 블록형의 병렬준비행렬로서 직접스파스해법과 복합된 다중색채 블록 SOR을 제안한다. Laplacian 행렬에서 SOR방법은 다중색채기법과 복합사용할 경우 수렴율이 저하되지 않는 것으로 알려져있다[1]. 대부분의 시간이 대각선 블록의 해법에 사용되므로 직접스파스해법의 속성에 따라 확장가능성이 기대된다. 우리는 이 결과를 다른 4가지 준비행렬(다중색채 ILU(0), wavefront ILU(0), SPAI, SSOR)과 비교분석한다. 시험행렬은 크기가 1024x1024까지의 2차원 유한차분행렬들이다. 128 node를 가진 Kordic center의 CRAY-T3E에 구현되었으며 MPI 라이브러리를 이용하였다. 결론은 유한차분행렬에서 다중색채 블록 SOR 준비행렬의 성능이 확장가능하면서 가장 수렴율이 우수했다.

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

Sang back Ma†

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. 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. CRAY-T3E with 128 nodes was used. MPI library was used for interprocess communications. The results show that Multi-Color Block SOR is scalable and gives the best performances.

키워드 : Sparse, Parallel, Preconditioner, CRAY-T3E, Scalable

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-

* 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.
† 종신회원 : 한양대학교 전자컴퓨터공학부 교수
논문접수 : 2001년 6월 30일, 심사완료 : 2001년 8월 21일

(Conjugate Gradient) method.

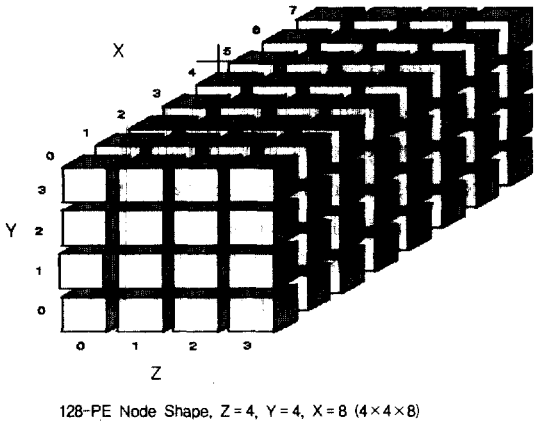
Wavefront or *level scheduling*[2] technique is still another way to achieve a parallelism while preserving the initial ordering. It does not suffer from the deterioration of the rate of convergence as with multi-coloring, but the maximum parallelism is determined by the lengths of the wavefront, which 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[3]

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.



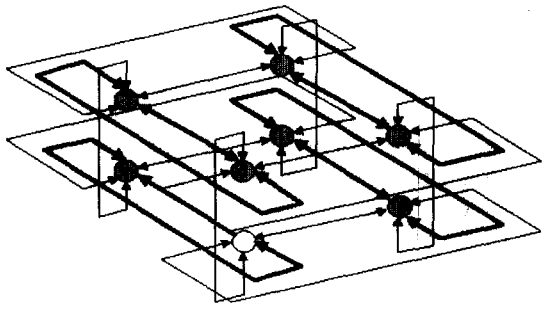
128-PE Node Shape, Z=4, Y=4, X=8 (4×4×8)

(Figure 1) CRAY-T3E PEs

2. Multi-coloring and Wavefront reordering

2.1 ulti-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.



Three-dimensional Torus Network

(Figure 2) CRAY-T3E Network

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.[1]

〈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[4] 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 [5]

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

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[6] 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 [7], 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 ≠ 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.

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. $(u_{i+1}^{GS})_j = D_j^{-1}(b - \sum_{k=1}^{q-1} E_{j,k} u_i)$
7. $u_{i+1} = u_i + \omega^* (u_{i+1}^{GS} - u_i)$.
8. endif
9. Endfor
10. Endfor
11. Endfor

(Algorithm 2.1) Multi-Color Block SOR

The u_i^{GS} is the i-th update of Block Gauss-Seidel iteration. If ω =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.

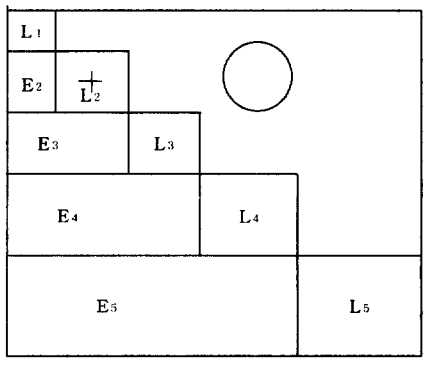
For example, let us assume that we have an ILU(0) factorization for the 5-point Laplacian for the Poisson equation, and for a preconditioner we need to solve

$$Lz = y,$$

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 (Figure 3), 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 [2]



(Figure 3) Block partitioning for L

3. SPAI Preconditioner

We look for M such that $\| AM-I \|_F^2$ is minimized where $\| \|_F$ stands for the Frobenius norm. Then since

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

finding such M separates into n independent least squares problems

$$\min_{m_k} \| Am_k - e_k \|_2 , k=1,\cdots n, \quad (3)$$

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 [3].

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 $\widehat{A} = A(I,J)$, $\widehat{e_k} = e_k(I)$. Find the QR decomposition of $\widehat{A}$,

$$\widehat{A} = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \quad (4)$$

Then, solve

$$\min_{\widehat{m_k}} \| \widehat{A}\,\widehat{m_k} - \widehat{e_k} \|_2 \quad (5)$$

and the residual $\gamma = A(.,J)\widehat{m_k} - ek$.

While $\| \gamma \|_2 > \tau$

(c) Set K equal to $\{ l \mid \gamma (l) \neq 0 \}$.

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

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

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

(g) Determine the new indices $\widetilde{I}$ and update the QR decomposition accordingly. Then, solve the new least squares problem, compute the new residual $r = A m_k - e_k$ and set $I = I \cup \widetilde{I}$, $J = J \cup \widetilde{j}$. For further details see [3].

4. ILU(0) factorization

Meijerink and Van. der Vorst[8] introduced a so-called Incomplete LU(I LU) preconditioner for symmetric matrices. The following is a modification of the original I LU for nonsymmetric matrices, as described in [9]. 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.

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

(Algorithm 4.1) ILU Factorization

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

$$(D-\omega E)x_{i+\frac{1}{2}} = ((1-\omega)D+\omega F)x_i + \omega b \quad (6)$$

$$(D-\omega F)x_{i+1} = ((1-\omega)D+\omega E)x_{i+\frac{1}{2}} + \omega ba$$

Endfor

5. Experiments

5.1 Test problems

● **Problem 1** Elman 's problem [9]

$$-(bu_x)_x-(cu_y)_y+(du)_x+du_x+(eu)_y+eu_y+fu=g \quad (7)$$

$$\Omega = (0,1)\times(0,1)$$
$$u = 0 \ on \ \delta\Omega$$

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

$$-\varepsilon\triangle u+\cos\alpha u_x+\sin\alpha u_y = f, \qquad (8)$$
$$\Omega = (0,1)\times(0,1)$$
$$u = 0 \ on \ \delta\Omega$$

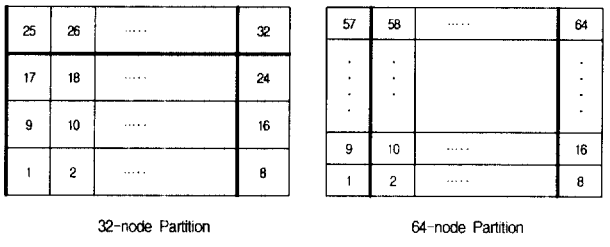
We have set $\varepsilon=0.0001$ and $\alpha=15$degree. Upwind scheme was adopted for the convection term. 5-point approximation was used for 2-nd order derivatives in Problem 1, and 9-point Laplacian for Problem 2.

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 (Figure 4)



(Figure 4) Mapping between a square domain and processors

5.3 Results

GMRES(10)[10] and BICGSTAB[11] were used as the outer iterative method. As for the manipulation and storing of the matrices the CSR(Compress Sparse Row) format[12] 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 $\gamma = 50, \beta = 1, \varepsilon = 0.1, \alpha = 15$, 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 [7]. 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.

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. 2-16. '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 is 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. BICGSTAB and GMRES(10) do not show any major difference in the above conclusions, except that the number of iterations for BICGSTAB is about half of that of GMRES(10).

〈Table 2〉 Problem1 with BICGSTAB 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	0.25/6
SSOR-AVEFRONT	1.42/62	1.04/62	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, τ = 0.2	2.82/64	1.69/62	1.23/64	1.32/62	1.22/62
SPAI, τ = 0.4	4.28/144	2.41/142	1.78/137	1.50/135	4.14/153

〈Table 3〉 .Problem1 with BICGSTAB 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-AVEFRONT	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, τ = 0.2	21.96/137	12.48/171	6.90/136	4.80/137	4.35/141
SPAI, τ = 0.4	29.16/276	16.71/296	9.10/281	5.30/264	8.22/271

〈Table 4〉 Problem1 with BICGSTAB N=512x512

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

〈Table 5〉 Problem1 with BICGSTAB N=1024x1024

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

〈Table 6〉 Problem2 with BICGSTAB 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-AVEFRONT	1.63/65	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	0.93/446	1.41/46
SPAI, τ = 0.2	3.57/68	2.04/68	1.41/68	1.55/68	2.22/68
SPAI, τ = 0.4	5.40/194	3.51/194	2.00/201	2.29/211	3.62/204

〈Table 7〉 Problem2 with BICGSTAB 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-AVEFRONT	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, τ = 0.2	28.08/141	14.82/141	8.24/141	5.13/141	4.91/141
SPAI, τ = 0.4	41.86/388	24.09/426	11.28/376	7.08/389	7.87/411

〈Table 8〉 Problem2 with BICGSTAB N=512x512

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

〈Table 9〉 Problem2 with BICGSTAB N=1024x1024

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

〈Table 10〉 Problem1 with GMRES(10) N=128x128

	P=4	P=8	P=16	P=32	P=64
	CPU time/Iterations				
MC-BSOR(2)	1.93/3	0.57/3	0.26/3	0.14/3	0.20/3
SSOR-AVEFRONT	1.14/31	0.83/31	0.86/30	1.09/31	1.49/31
ILU(0)-WAVEFRONT	0.79/25	0.64/26	0.72/27	0.86/25	1.27/26
ILU(0)-MULTICOL	0.76/25	0.56/27	0.53/27	0.55/25	0.78/26
SPAI, τ = 0.2	2.26/32	1.35/31	0.99/32	1.06/31	0.98/31
SPAI, τ = 0.4	3.42/72	1.93/71	1.42/68	1.20/68	3.31/76

〈Table 11〉 Problem1 with GMRES(10) N=256x256

	P=4	P=8	P=16	P=32	P=64
	CPU time/Iterations				
MC-BSOR(2)	14.64/3	4.91/3	1.98/4	0.78/4	0.51/5
SSOR-AVEFRONT	8.98/63	5.16/61	3.94/66	3.24/60	3.82/60
ILU(0)-WAVEFRONT	5.64/54	3.84/54	2.82/54	2.57/52	3.30/53
ILU(0)-MULTICOL	6.32/54	3.58/54	2.29/54	1.69/51	1.86/53
SPAI, τ = 0.2	17.57/69	9.98/65	4.62/68	3.84/69	3.48/70
SPAI, τ = 0.4	23.33/138	13.37/148	7.28/141	4.24/132	6.58/136

〈Table 12〉 Problem1 with GMRES(10) N=512x512

	P=4	P=8	P=16	P=32	P=64
	CPU time/Iterations				
MC-BSOR(2)	×	×	16.8/6	6.1/6	2.88/8
SSOR-AVEFRONT	×	34.41/129	19.8/128	14.1/129	12.21/128
ILU(0)-WAVEFRONT	×	25.64/117	15.13/117	10.5/112	9.72/113
ILU(0)-MULTICOL	×	24.34/115	13.76/115	7.91/114	6.16/112
SPAI, τ = 0.2	171.4/165	76.46/156	42.7/156	24.02/151	15.4/159
SPAI, τ = 0.4	196.79/286	93.79/269	52.93/286	42.31/281	18.96/277

〈Table 13〉 Problem1 with GMRES(10) N=1024x1024

	P=4	P=8	P=16	P=32	P=64
	CPU time/ Iterations				
MC-BSOR(2)	×	×	×	54.72/9	20.16/9
SSOR-AVEFRONT	×	×	×	85.18/267	59.05/282
ILU(0)-WAVEFRONT	×	×	×	64.13/241	42.95/223
ILU(0)-MULTICOL	×	×	×	53.67/234	32.82/246
SPAI, τ = 0.2	×	×	×	238.85/320	105.64/325
SPAI, τ = 0.4	×	×	×	253.14/687	127.36/627

〈Table 14〉 Problem2 with GMRES(10) N=128x128

	P=4	P=8	P=16	P=32	P=64
	CPU time/ Iterations				
MC-BSOR(2)	3.57/2	1.04/2	0.44/2	0.32/3	0.27/3
SSOR-AVEFRONT	1.30/33	1.23/34	1.34/33	1.57/33	2.47/33
ILU(0)-WAVEFRONT	0.74/19	0.65/21	0.78/21	0.91/21	1.55/22
ILU(0)-MULTICOL	0.76/19	0.62/21	0.62/21	0.74/22	1.13/23
SPAI, τ = 0.2	2.86/34	1.63/34	1.13/34	1.24/34	1.78/34
SPAI, τ = 0.4	4.32/97	2.81/97	1.60/101	1.83/106	2.90/102

〈Table 15〉 Problem2 with GMRES(10) N=256x256

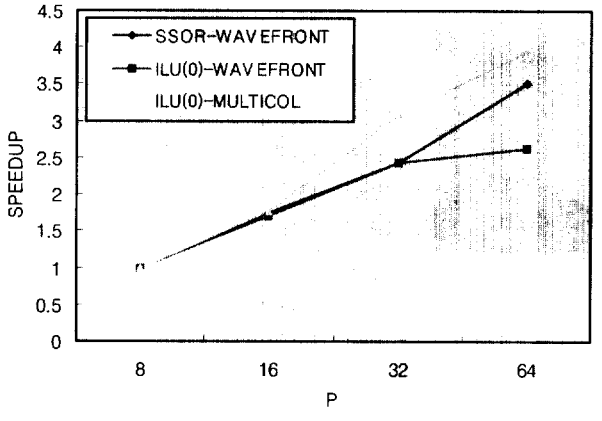
	P=4	P=8	P=16	P=32	P=64
	CPU time/ Iterations				
MC-BSOR(2)	31.6/2	9.92/2	3.50/2	1.24/2	0.58/2
SSOR-AVEFRONT	11.7/64	6.29/62	5.01/63	4.76/64	6.03/65
ILU(0)-WAVEFRONT	5.66/37	3.45/40	2.68/38	2.79/42	3.61/41
ILU(0)-MULTICOL	5.69/37	3.30/41	2.30/40	1.91/39	6.58/78
SPAI, τ = 0.2	22.46/71	11.86/71	6.59/71	4.10/71	3.93/71
SPAI, τ = 0.4	33.49/194	19.27/213	9.02/188	5.66/195	6.30/101

〈Table 16〉 Problem2 with GMRES(10) N=512x512

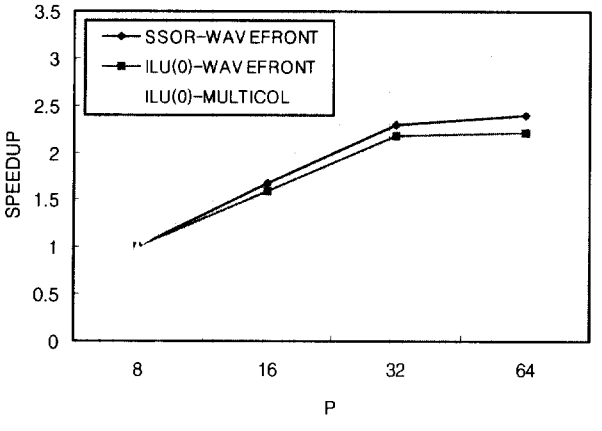
	P=4	P=8	P=16	P=32	P=64
	CPU time/ Iterations				
MC-BSOR(2)	×	×	31.20/2	10.56/2	3.84/2
SSOR-AVEFRONT	×	42.3/119	25.14/116	18.38/120	17.62/122
ILU(0)-WAVEFRONT	×	21.6/74	13.54/74	9.93/76	9.77/75
ILU(0)-MULTICOL	×	20.75/74	11.35/71	7.74/75	6.57/78
SPAI, τ = 0.2	180.88/141	92/143	47.22/141	26.06/143	16.18/140
SPAI, τ = 0.4	276.14/408	138.37/408	68.9/392	35.2/375	24.22/390

〈Table 17〉 Proble3m2 with GMRES(10) N=1024x102

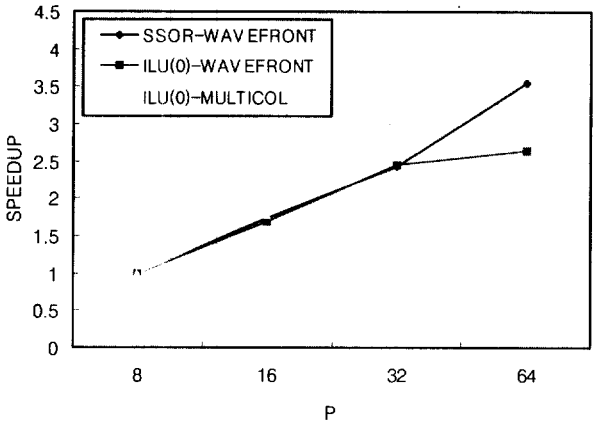
	P=4	P=8	P=16	P=32	P=64
	CPU time/ Iterations				
MC-BSOR(2)	×	×	×	SLOW	33.92/3
SSOR-AVEFRONT	×	×	×	102.53/236	70.91/230
ILU(0)-WAVEFRONT	×	×	×	49.12/135	36.96/140
ILU(0)-MULTICOL	×	×	×	44.56/143	27.94/145
SPAI, $\tau = 0.2$	×	×	×	273.23/268	127.46/268
SPAI, $\tau = 0.4$	×	×	×	353.61/1000	161.98/862



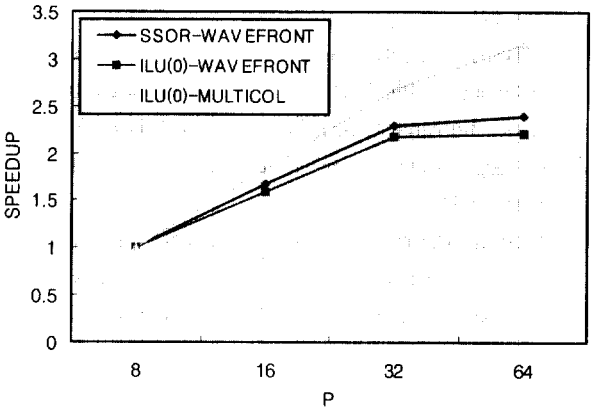
(Figure 5) Speedup of Problem1 with BICGSTAB, N=512x512



(Figure 6) Speedup of Problem2 with BICGSTAB, N=512x512



(Figure 7) Speedup of Problem1 with GMRES(10), N=512x512



(Figure 8) Speedup of Problem2 with GMRES(10), N=512x512

6. Summary

▮ Unlike in the symmetric case the convergence rate of the BICGSTAB and GMRES(m) methods do 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 and GMRES(m) method in the multi-coloring order do not deteriorate, compared to that of the naturalordering. In future we might need more research to explain this phenomenon.

References

[1] 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.

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

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

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

620-632.

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

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

[7] 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.

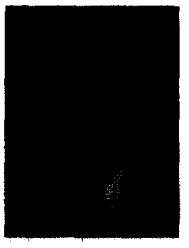
[8] J.~A. Meijerink and H.~A. van~der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric {M}-matrix," *Math. Comp.*, Vol.31, 1977, pp.148-162.

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

[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] 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.

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



마 상 백

e-mail : sangback@cse.hanyang.ac.kr

1978년 서울대학교 계산통계학과 졸업(학사)

1978년~1983년 국방과학연구소 연구원

1987년 미국 미네소타 주립대학 수학과 석사

1993년 동대학 전자계산학과 박사

1994년~1996년 한양대학교 공학대학 전자계산학과 전임강사

1996년~2001년 한양대학교 전자컴퓨터 공학부 조교수

2001년~현재 한양대학교 전자컴퓨터 공학부 부교수

관심분야 : 수치해석, 암호학, 병렬처리