Parallel Poisson Solver FAGECR—Implementation and Performance Evaluation on PAX Computer

TSUTOMU HOSHINO*, YOSHIYUKI SATO*† and YUKIKO ASAMOTO**

Fast Poisson equation solver "FAGECR" was implemented on the PAX computer, a nearest-neighbor-mesh connected parallel computer. The algorithm follows basically the FACR, Fourier-Analysis-Cyclic-Reduction method, by R. W. Hockney. A special parallel algorithm was employed to solve the tridiagonal equations, that combines Gaussian elimination in each processor and cyclic reduction over all processors.

Execution times were measured by the hardware timers. The computation speed is generally faster, by approximately 10 times, than that observed for optimally accelerated SOR method with odd-even ordering.

The performance was analyzed to get "scaling law", expressing the time as a function of problem size and number of processors. The scaling law can be used to predict the performances that would be obtained in cases with larger number of processors and problem sizes.

1. Introduction

Poisson equation is a very basic equation used in scientific calculations, whose quick solution is required in the electro-magnetic problems, fluid dynamics, and other major applications solving the potential problems. The Poisson equation often appears in the deepest loop in the iterative schemes, so that many quick schemes have so far been developed, among which the FACR, Fourier-Analysis-Cyclic-Reduction method [1] is one that gives $O(N \log N)$ time complexity in solving 2-dimensional Poisson equations. The FACR scheme consists of several numerical techniques, such as cyclic-reduction, FFT, tridiagonal linear equation solver, inverse FFT, and cyclic-expansion in the last stage. Parallelizing the FACR method was studied [2] assuming the use of a vector processor as well as an array of processors.

Although an algorithm itself is independent from a machine on which the algorithm is implemented, the performance finally realized on the machine is heavily dependent on the method of mapping the variables and schemes on the processors and memories. Since the inter-processor communication overhead and idling loss of processors are two major causes that degrade the performance of implemented parallel algorithm, the best mapping method should also be determined in relation to the architecture of inter-processor and memory connection.

*Institute of Engineering Mechanics, University of Tsukuba, Tsukuba-shi, Ibaraki-ken 305, Japan
*†Toshiba Corp., Fuchu Works, Toshiba-cho, Fuchu-shi, Tokyo 183, Japan
**Fuji Photofilm Co., Ltd. Development Center, Miyanodai, Kaisei-machi, Ashigarakami-gun, Kanagawa 258, Japan

This paper reports how FACR method was implemented on the PAX computer, a nearest-neighbor-mesh connected processor array, and what were the major factors that affected the performance obtained in several mapping scheme. Although the algorithm studied in this paper is essentially the same as the serial algorithm FACR, (except that introduced in solving a tridiagonal linear equations, already proposed in Ref. [5]), interrelation between algorithm, machine architecture, and the mapping scheme are an important topics that should be studied to realize a fast implementation of an algorithm, effectively used in scientific applications.

We implemented the algorithm on the PAX computer [3], which is an NNM (nearest-neighbor-mesh) connected array of processing units (PUs). The latest version of PAX, PAX-64J currently has 32 PUs [4] installed. Each PU consists of a 16-bit microprocessor, local RAM, 4 communication shared memories, and several registers to be used for controlling the array. The whole PU array has a shape of torus, where both sides are connected, and so do the top and bottom of the array. Each row and column of the PU array can work as a ring array of PUs.

The simplest mapping, frequently used in the PAX applications, is the direct mapping, where each PU works on the physical subregion (called PU-subregion, hereafter) that is allocated to the PU by directly projecting the whole physical space onto the PU array. If we distribute the variables simply following this direct mapping and implements the FACR scheme as it is, the tridiagonal equation must be solved by a set of ring array of PUs, where inter-PU communication may take a major portion of the execution time. Parallel implementation and numerical scheme have to be one that

reduces the communication overhead in this tridiagonal solver.

The triadiagonal solver that we used in this study is called GECR, Gaussian-Elimination-Cyclic-Reduction [5] scheme, which was proved effective in solving the tridiagonal equations by the PAX computer in the Navier-Stokes equation solver using Beam-Warming Implicit method [6]. The GECR removes this crucial problem; how we solve tridiagonal equations on the 1-dimensional PU array of PAX, with a small communication overhead.

The execution time was measured and condensed into what we call "scaling law", a functional expression of execution time with several parameters, such as size of problem, size of PU array, and other important hardware parameters, as well as the mapping schemes. Based on this scaling law, the performance can be discussed for larger size of problems implemented on the larger size of PU arrays.

For reader's convenience, the meaning of abbreviations of algorithms used in this paper is summarized in Table 1, together with associated explanations/comments.

2. Poisson Equation and Solution by FACR Method [1]

Poisson equation in rectangular 2-dimensional space with periodic boundary condition,

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f, \quad (0\leqq x\leqq X, 0\leqq y\leqq Y) \tag{1}$$

$$u(0, y)=u(X, y), \quad u(x, 0)=u(x, Y) \tag{2}$$

$$\frac{\partial u}{\partial x}(x=0)=\frac{\partial u}{\partial x}(x=X) \quad \frac{\partial u}{\partial y}(y=0)=\frac{\partial u}{\partial y}(y=Y) \tag{3}$$

Table 1 Summary of algorithms.

Abbreviations	Explanations
FACR(L)	Serial algorithm for solving Poisson equation, with L-stage CR(Cyclic-Reduction), proposed by *R.* Hockney [1]. This consists of CR, FFT, a tridiagonal solver, inverse FFT, and expansion stage of CR.
FAGECR(L)	Parallel version of FACR(L), implemented on PAX. This consists of PARACR, FFT, GECR, inverse FFT, and expansion stage of PARACR.
GECR	Gaussian Elimination Cyclic Reduction method for solving tridiagonal linear equations. This consists of GE(Gaussian Elimination), PARACR, and backward substitution of GE.
PARACR	Parallel version of CR, studied by *R.* Hockney [2].
FAGECR1D(L)	FAGECR(L) scheme mapped onto 1-dimensional PU array of PAX computer.
FAGECR2D(1)	FAGECR(1) scheme mapped onto 2-dimensional PU array of PAX computer.

is discretized into 5-point formula, as follows,

$$u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}-4u_{i,j}=f_{i,j} \tag{4}$$

assuming that

$$x=i,\ y=j,\ (i=1, 2, \ldots, N_1),\ (j=1, 2, \ldots, N_2), \tag{5}$$

which is further expressed in a vector form, as follows.

$$U_{j-1}+AU_j+U_{j+1}=F_j \tag{6}$$

$$U_j=(u_{1,j}, u_{2,j}, \ldots, u_{N_1,j})' \tag{7}$$

$$F_j=(f_{1,j}, f_{2,j}, \ldots, f_{N_1,j})' \tag{8}$$

$$A=\begin{bmatrix} -4 & 1 & & & & & 1 \\ 1 & -4 & 0 & & & & \\ 0 & 1 & -4 & 1 & & 0 & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & 0 & & & & & \\ & & & & \cdot & \cdot & \cdot \\ & & & & & -4 & 1 \\ 1 & & & & & 1 & -4 \end{bmatrix} \tag{9}$$

The FACR method consists of the following 5 stages.

Stage I. Applying L-times cyclic-reduction scheme in y-direction (in direction of increasing j) to vector equation (6), we have a reduced set of tridiagonal vector equation,

$$U_{j-\$L}+A^{(L)}U_j+U_{j+\$L}=F_j^{(L)} \tag{10}$$

where,

$$A^{(L)}=2I-(A^{(L-1)})^2,\ A^{(0)}=A, \tag{11}$$

$$F_j^{(L)}=F_{j-\$(L-1)}^{(L-1)}-A^{(L-1)}F_j^{(L-1)}+F_{j+\$(L-1)}^{(L-1)}, \tag{12}$$

$$F_j^{(0)}=F_j.$$

The functional notation $\$(X)=2^X$ is used throughout the paper, where $\$L=2^L$, $\$(L-1)=2^{(L-1)}$, and $0\leqq L\leqq \log_2 N_2$.

Stage II. Applying FFT in x-direction, i.e. increasing direction of i, to Eq. (10), we obtain the following N_1 tridiagonal systems, each of which is a set of $N_2/\$L$ equations in $N_2/\$L$ variables and is decoupled with respect to k, the transformed variable of x.

$$u_{k,j-\$L}+a_k^{(L)}u_{k,j}+u_{k,j+\$L}=f_{k,j}^{(L)} \tag{13}$$

where

$$a_k^{(L)}=2-(a_k^{(L-1)})^2, \tag{14}$$

$$a_k^{(0)}=-4+2\cos(\pi k/N_1), \tag{15}$$

and $k=0, 1, 2, \ldots, N_1-1$, $j=\$L, 2\$L, 3\$L, \ldots, N_2$.

Stage III. Solving the N_1 triadiagonal systems (13) by any algorithm such as Gaussian-elimination or cyclic-

reduction method, we obtain the Fourier-transformed solution U.

Stage IV. Applying inverse-FFT to U, we obtain

$$u_{i,j},\ (i=1, 2, \ldots, N_1,\ j=\$L, 2\$L, 3\$L, \ldots, N_2). \tag{16}$$

Stage V. Trancing backward the cyclic-reduction scheme, from Eq. (10) to Eq. (12), we finally expand the solution to all variables,

$$u_{i,j},\ (i=1, 2, \ldots, N_1, j=1, 2, \ldots, N_2). \tag{17}$$

Equation 10 is a set of vector forms of tridiagonal systems, each of which is a set of tridiagonal equations in the variables previously eliminated in the cyclic reduction scheme in stgage I, i.e.

$$A^{(s)}U_j=F_j^{(s)}-U_{j-\$s}-U_{j+\$s} \quad \text{for} \quad s=1, 2, \ldots, L, \tag{10'}$$

and

$$A^{(s)}=-\prod_{r=1}^{\$s} B^{(r)},\ B^{(r)}=(A-\beta_r I),$$
$$\beta_r=2\cos[(2r-1)\pi/\$(r-1)], \tag{18}$$

where the matrix $A^{(s)}$ is factorized by a sequence of tridiagonal matrices $B^{(r)}$. Therefore the expansion is essentially to solve repeatedly the triadiagonal linear equations with a form $B^{(r)}V_r=V_{r-1}$.

3. Mapping Scheme and Optimum Cyclic Reductions

One of the crucial considerations in implementing the parallel algorithms on a PAX computer is the PU-mapping: whether mapping of 2-dimensional physical space should be made onto the 1-dimensional PU array or 2-dimensional torus array as the PAX is originally designed. Generally a 1-dimensional PU array has a drawback that, when the problem size becomes large, each PU has to install memory space proportional to the problem size, and the data communication across the array takes time proportional to the size of the array.

The 1-dimensional mapping, however, makes program simpler, and, in this FACR scheme, the vector-cyclic-reduction stage requires less inter-PU communication in deep reductions than that in the 2-dimensional mapping.

These questions will be answered, though partly, in this paper, where we implement the FACR scheme in two ways: onto 1-dimensional PU array with optimum number of cyclic-reduction levels, i.e. FAGECR1D(L) and onto 2-dimensional PU array with single reduction stage, i.e. FAGECR2D(1).

Another interesting point to be studied is what the optimum number of levels of the vector-cyclic reduction is. The number of unknown variables to be solved by FFT and tridiagonal solvers is reduced to 1/2 as the vector cyclic-reduction proceeds by 1 level. Though the vector cyclic-reduction saves time in FFT and tridiagonal solutions, it costs extra-time in the cyclic-expansion stage. An optimum level is determined by the trade-off between the saving and the costing.

4. Parallel Implementation of FACR onto PAX

The mapping of the 2-dimensional physical space can be made in two ways: FAGECR2D where the projection was made onto 2-dimensional PU array as shown in Fig. 1(a), and FAGECR1D where the projection was made onto 1-dimensional PU array as shown in Fig. 1(b), both of which were tested. Fig. 2 illustrates how the scheme eliminates the variables and links, and how it expands the solution in two adjacent PU-subregions.

4.1 Parallelization of Vector Cyclic Reduction Stage I

FAGECR2D scheme multiplies matrix A, L-times, on vector U_j, where inter-PU data transfer occurs in x- and y-directions, because, at each multiplication of matrix A, the data distributed over the space, single-layer distant from each point, are involved, and, after L-times multiplications, the data distant by L space points from the boundary of the PU-subregion have to be transferred.

In FAGECR1D scheme, similar data transfer during the multiplication of matrix A occurs, but only in y-direction. By restricting the space size in y-direction in a PU greater than or equal to $\$L$, the data transfer is not beyond the nearest-neighbor PU.

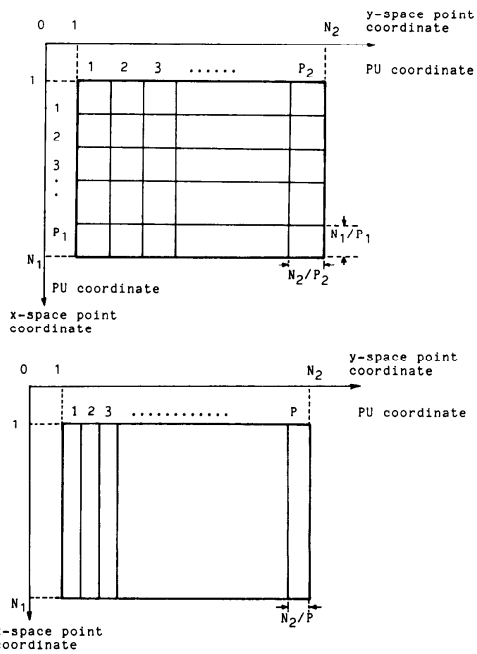

Fig. 1 Two methods of mapping of physical space onto the PU array: (a) 2-dimensional, (b) 1-dimensional.

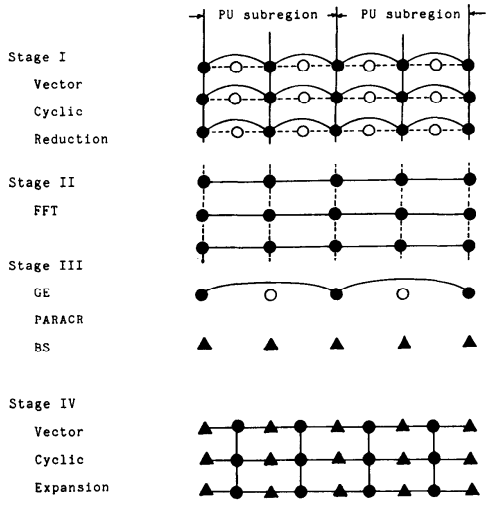

Fig. 2 Illustration of FAGECR scheme, where ●: unknown variables, ▲: known variables, ○: eliminated variables, ---: eliminated links ——: effective links.

4.2 Parallelization of FFT and Inverse FFT in Stages II and IV

Conventional Cooley-Tukey's algorithm was used. Fourier transform is made in x-direction. In FAGECR2D, the right-hand-side vector f has to be butterfly-transfered among PUs, while in FAGECR1D, all computation is kept within each PU and no data is transfered among PUs.

4.3 Parallelization of Solving Tridiagonal Equation in Stage III

As mentioned in the introduction, one of the crucial point in the parallel implementation of the FACR scheme exists in stage III, where N_1 tridiagonal systems, each of which is a set of N_2 equations in variables allocated in y-direction, are to be solved by a single set of ring array of P PUs in FAGECR1D, and by P_1 sets of ring arrays of P_2 PUs in FAGECR2D.

The GECR method [5] has been developed for this purpose, where:

i) A set of tridiagonal equations in $M\,(=N_2)$ unknown variables, $(u_1, u_2, \ldots, u_M)$, are partitioned into P blocks by P PUs, as shown in Fig. 3, and the variables in each block, such as $u_2, u_3, \ldots, u_{(M/P)}$, are Gaussian−eliminated to get a reduced set of tridiagonal equations in P variables, $(R_1, R_2, \ldots, R_P)$, allocated to the P boundaries of the PU-subregions, where the periodic boundary condition is imposed. The time complexity of calculation in this phase I is $O(M/P)$ with no data transfer.

ii) This set of triadiagonal equations in P variables is solved by the PARACR, the parallel cyclic reduction scheme [2] with the time complexity $O(\log P)$ of calculation and the complexity $O(P)$ of the data transfer between adjacent PUs.

iii) The backward substitution of the boundary variables to the interior variables are completed with the time complexity $O(M/P)$ with no data transfer.

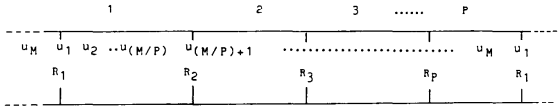

Fig. 3 Partitioning of physical space into PU-subregions in the GECR method solving tridiagonal equations.

It must be noted that the GECR scheme is "consistent" [7] in the sense that the time complexity of GECR $O(M)$ with respect to the problem size M is equal to that of the serial counterpart, the Gaussian elimination. This assures that the GECR is no slower than the serial scheme in the limit of very large values of M.

4.4 Parallelization of Vector Cyclic-Expansion in Stage V

Solution is expanded to the $N_1*(N_2/P)*(1-1/\$L)$ unknown variables, once eliminated during the vector cyclic reduction, by solving the $(N_2/P)*(1-1/\$L)$ tridiagonal systems, each of which is a set of N_1 equations. This process is executed in parallel in each PU in FAGECR1D scheme. At the beginning of the stage, variables located in single layer from the boundary must be transferred between the adjacent PUs. While in FAGECR2D scheme, the tridiagonal systems are solved by applying the GECR scheme in x-direction, with the time complexity already mentioned, and with the same data transfer as that in FAGECR1D, at the beginning of this stage.

5. Performance Measurement and its Scaling

The FAGECR1D and FAGECR2D schemes were actually implemented on a PAX-64J computer, and the performance was measured to determine if one can extend the present performance to the cases with greater problem sizes and PU arrays.

5.1 Measured Timing and Scaling Law

A Poisson equation was solved in a rectangular region with the sizes N_1 and N_2, and with two point sources of intensities $+1$ and -1, respectively, at the symmetrical points on a diagonal line of the rectangular region, as shown in Fig. 4. The boundary condition is periodic in both x- and y-directions.

Execution times are listed in Table 2 and Table 3. Because of the limitation of memory space in a single PU for both data and program, only the single cyclic-reduction (i.e. FAGECR2D(1)) was programmed. The FAGECR1D(L) schemes was also tested only for those cases that have at least one reduced variable per PU after the vector-cyclic reduction.

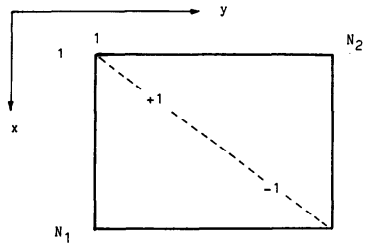

Fig. 4 Rectangular physical space with periodic boundary condition, and two point sources, in which Poisson equation is solved.

Table 2 Measured execution times (in seconds) of FAGECR1D(L) scheme with using $P=8$ PUs.

Number of Stage L	Region Sizes N_1*N_2 16*128	32*32	32*64	32*128	64*64
0	—	0.3132	—	—	—
1	0.3449	0.3004	0.4410	0.7468	0.9456
2	0.3460	—	0.4378	0.7226	0.9083
3	0.3934	—	—	0.7852	—

Table 3 Measured execution times (in seconds) of FAGECR2D scheme with using $P=4*8$ PUs, in comparison with methods of double FFT, odd-even SOR.

Schemes	Region Sizes N_1*N_1 32*32	64*64	128*128
FAGECR2D(0)	0.136	0.431	—
FAGECR2D(1)	0.113	0.318	1.021
Double FFT	0.172	0.667	2.779
Odd-even SOR	0.366	2.501	17.776

Execution times are also measured by implementing the other representative methods, such as FAGECR(0) with no vector reduction, double FFT method applying 1-dimensional FFT in both x- and y-directions, odd-even SOR method with optimum accerelation. The FAGECR(0) and the double FFT methods were chosen to identify the effect of vector-cyclic reduction in FAGECR(L). The odd-even SOR method was chosen because it is best-fitted algorithm with PAX's nearest-neighbor-mesh architecture and the comparison would indicate the architectural preference of FAGECR. The double FFT and SOR methods are simpler in implementation than FAGECR(L), and the comparison may indicate the gain in performance of FAGECR over the complexity paid in its implementation. Moreover these methods are representative in the sense that they are frequently used in many scientific applications, though the comparison does not exhaust all the algorithms proposed for Poisson solving, especially the one with lower complexity such as the multigrid method [8].

In our method of evaluating the performance, the timing expression, what we call "scaling law", is derived from these measurements, and performance with different size parameters is discussed by applying those parameters to the scaling law expression. For the FAGECR1D scheme, the scaling law is given as shown in Table 4. Note that only the leading terms with the highest order complexity are shown, which do not regenerate the measured times in Tables 2 and 3.

Table 4 Scaling law of execution times in FAGECR1D(L). Only the leading terms are shown.

Stages	Net Calculation	Overhead
Vector Cyclic-Reduction	$a_1 nL$	$b_1 N_1 L$
FFT	$a_2(n/\$L)\log N_1$	0
GECR:		
Gaussian-Elimination	$a_3(n/\$L)$	$b_2 N_1$
Cyclic-Reduction	$a_4 N_1 \log P$	$b_3 N_1 P$
Backward-Subsitution	$a_5(n/\$L)$	$b_4 N_1$
Inverse FFT	$a_6(n/\$L)\log N_1$	0
Vector Cyclic-Expansion	$a_7 nL$	$b_5 N_1$

$a_1=107$, $a_2=65$, $a_3=340$, $a_4=323$, $a_5=190$, $a_6=115$, $a_7=25$, $b_1=14$, $b_2=122$, $b_3=206$, $b_4=106$, $b_5=38$ (in unit of μsec)

Table 5 Scaling law of execution times in FAGECR2D(1). Only the leading terms are shown.

Stages	Net Calculation	Overhead
Vector Cyclic-Reduction	$a_1 n$	$b_1 n_1 + b_2 n_2$
FFT	$a_2 n \log N_1$	$b_3 n P_1$
GECR:		
Gaussian-Elimination	$a_3 n$	$b_4 n_1$
Cyclic-Reduction	$a_4 n_1 \log P_2$	$b_5 P_2$
Backward-Substitution	$a_5 n$	$b_6 n_1$
Inverse FFT	$a_2 n \log N_1$	$b_3 n P_1$
Vector Cyclic-Expansion by GECR:		
Gaussian-Elimination	$a_6 n$	$b_7 n_2$
Cyclic Reduction	$a_7 n_1 \log p_2$	$b_8 P_2$
Backward-Substitution	$a_8 n$	$b_9 n_2$

$a_1=27$, $a_2=64$, $a_3=155$, $a_4=291$, $a_5=95$, $a_6=53$, $a_7=43$, $a_8=60$, $b_1=16$, $b_2=20$, $b_3=27$, $b_4=103$, $b_5=247$, $b_6=39$, $b_7=46$, $b_8=67$, $b_9=22$ (in unit of μsec)

The following symbols are used in the scaling law expression.

$n_1=N_1/P_1$, $n_2=N_2/P_2$: problem sizes per PU in x- and y-directions,

$n=n_1 n_2=(N_1 N_2)/P$: problem size in a PU,

All logarithms are binary, i.e. $\log X=\log_2 X$.

5.2 Optimum Reduction in FAGECR1D

The optimum number of reduction L^* is derived by differentiating the scaling law expression given in Table 4, and solving the equation: the differentiated expression $=0$ with respect to L.

The value of L^* thus obtained is shown in Table 6, where the continuous value of L is permitted. Though the true optimum L^{**} would be the nearest integer

Table 6 Optimum (continuous) value L^* for the level of vector cyclic reductions

Problem Size, N_1*N_2	P, Number of PUs used 4	8	16
32* 32	2.06	1.93	1.73
32* 64	2.13	2.06	1.94
32*128	2.16	2.13	2.06
64* 64	2.38	2.33	2.22
128*128	2.63	2.60	2.55
256*256	2.84	2.83	2.80

values of L^*, the trend observed in the measured execution times in Table 2 is consistent with these L^{**} obtained for $P=8$.

It can be observed, in Table 6, that L^* increases as the problem size per PU increases. Also the execution times become more insensitive to the change of L value, as the problem size per PU increases. Though the single level vector reduction is effective over no reduction scheme, the gain in the execution times by optimizing L is not great.

5.3 Extended Discussion on Performance

The performance scaling law is useful to predict the performance with larger sizes of PU arrays and problems. The performance is represented by the efficiency of parallel processing, that is defined by the ratio of the net calculation over the total execution including overhead and idling.

The realistic assumption in the performance prediction will be that both sizes N_1 and N_2 in x- and y-directions are equally expanded, proportional to P, that is $n=(N_1N_2)/P$ is proportional to P. Fig. 5 illustrates the performance of FAGECR1D(L) predicted for those cases with the combination of $L=1$, 2, and 3 and $N_1=N_2=8P$, $4P$, and $2P$. The asymptotic efficiency with very large P can be interpreted as the ratio of the net over the total complexities in the infinite limit of P. In this simulation case, the overhead term in the cyclic reduction in GECR has the order $O(P^2)$, higher than that in the net calculation $O(P\log P)$, so that the curves decline toward the infinite limit of P value. The dependence of the efficiency on L value is not so great as in the measured execution times.

Careful setup of the simulation cases is necessary for the comparison between the two mappings, FAGECR1D and FAGECR2D, both with a single reduction $L=1$. Here we assume that we are given a problem with the sizes N_1 and N_2, and a 2-dimensional PU-array with the sizes P_1 and P_2.

Further assumptions were made as follows:

1) $P=2^p$, where p is a positive integer.
2) $P=P_1P_2$, where $P_1=P_2$ or $P_1=2P_2$.
3) $N_1=n_1P_1$, $N_2=n_2P_2$, where n_1 and n_2 are independent parameters.

In 2-dimensional FAGECR2D mapping, each PU works on the subregion $n_1n_2=(N_1/P_1)(N_2/P_2)$, as shown in Fig. 1(a). In 1-dimensional FAGECR1D mapping, the same rectangular region with sizes N_1*N_2 is sliced by P PUs, as shown in Fig. 1(b), where each PU calculates the region with the size N_1 in x-direction and (N_2/P) in y-direction.

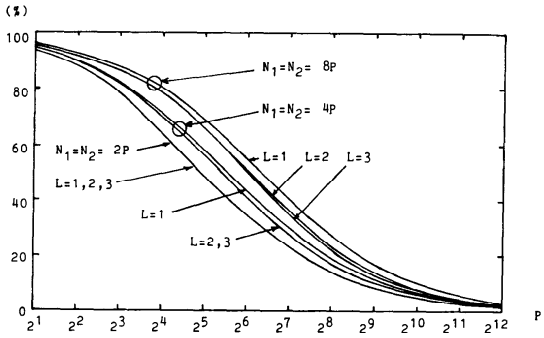

Fig. 5 Efficiency versus number of PUs, with changing the region sizes both in x- and y-directions, in proportional to number of PUs.

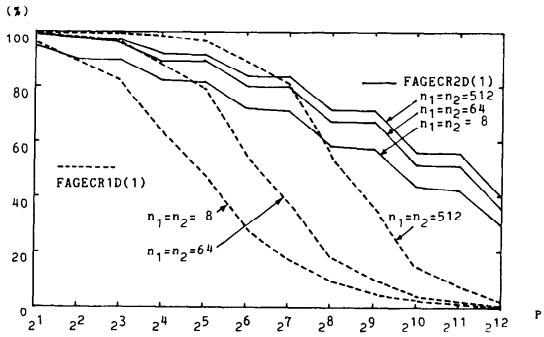

Fig. 6 Efficiency versus number of PUs, with changing the region sizes both in x- and y-directions. Poisson equation in physical space of the same sizes is solved by two mapping methods, FAGECR1D(1) and FAGECR2D(1), with the same number of PUs.

The performances are compared in Fig. 6, as p increases from 1 to 13, with 3 sets of parameters for n_1 and n_2. We can see that there is a range of p, $1\leqq p\leqq 7$, where the efficiencies are higher in 1-dimensional FAGECR1D than 2-dimensional FAGECR2D for cases with large problem sizes, such as the case with 512*512 space points per PU. Otherwise, the efficiencies are higher in 2-dimensional FAGECR2D than in 1-dimensional FAGECR1D.

6. Conclusions

The parallel implementation of FACR scheme solving Poisson equation in 2-dimensional rectangular region was made on the PAX-64J parallel computer. Two mapping methods, i.e. 1-dimensional FAGECR1D and 2-dimensional FAGECR2D mappings were tested. In the 1-dimensional mapping, the effect of the number of vector cyclic-reduction L on the execution times was

also studied. Though it was not possible to test all the combinations of parameters due to memory limitations, several conclusions were obtained as follows.

1) The FAGECR schemes developed here is faster by approximately 2 to 3 times than the double FFT method, and by approximately 10 times than odd-even SOR method.

2) The optimum value was observed in the number of reduction L^* in FAGECR1D(L), as it is in the serial or other vector implementation of FACR[2].

3) The gain in execution time and efficiency by optimizing L is not so great, though the single level reduction saves time by approximately $1/3$ over no reduction scheme.

4) By differentiating the scaling law expression for the execution timing, we can derive the optimum L value, which is near to the optimum number of reductions observed in the execution time measurement.

5) The optimum L value becomes large as the problem size per PU increases.

6) Assuming a linear expansion of the problem sizes in both x- and y-directions, FAGECR1D(L) displays a sharp decline of efficiency. Two-dimensional FAGECR2D(1) is generally superior to 1-dimensional FAGECR1D(1), except for cases with large number of space point per PU and processed with small number of PUs; for example the case with 512×512 space points per PU and processed with number of PUs less than 128.

Though the 2-dimensional mapping is generally better in performance than 1-dimensional mapping of FAGECR scheme, the 2-dimensional mapping has its own disadvantage, i.e. the programming of FAGECR2D(L), $L \geqq 2$ needs more effort than that of FAGECR1D(L). The 1-dimensional FAGECR1D(L) has also its own disadvantage: the local memory size per PU should be expanded in proportion to P, that will not be possible for a very large P, given a fixed memory capacity per PU.

The FAGECR is the fastest Poisson solver implemented on PAX at a moment. However, it does not exclude the needs for other methods with lower order of complexity, such as the multigrid method [8], to be compared in the performance as well as in the practical applicability, and to define the best solver or solvers for NNM machines. The practical solution would be to prepare several implementations of Poisson solvers, including SOR, multigrid, etc, and to assess before use, with the user's own parameters and conditions, where the scaling laws derived in this paper would be useful guidelines for the optimum choice of the algorithms.

References

1. HOCKNEY, R. W., RAPID ELLIPTIC SOLVERS, in "Numerical Method in Applied Fluid Dynamics", edited by B. Hunt (Academic Press, 1980), 1–48.

2. HOCKNEY, R. W. and JESSHOPE, C. R., Parallel Computers, (Adam Hilger, Bristol, 1981).

3. HOSHINO, T., KAWAI, T., SHIRAKAWA, T., HIGASHINO, J., YAMAOKA, A., ITO, H., SATO, T. and SAWADA, K., PACS, A Parallel Microprocessor Array for Scientific Calculations, *ACM Trans. Comput. Systems*, **1** (1983), 195–221.

4. HOSHINO, T., SHIRAKAWA, T. and TSUBOI, K., Mesh-connected parallel computer PAX for scientific applications, *Parallel Computing*, **5** (1987), 363–371.

5. HOSHINO, T., KAMIMURA, T., IIDA, T. and SHIRAKAWA T., Parallelized ADI Scheme using GECR (Gauss-Elimination-Cyclic-Reduction) Method and Implementation of Navier-Stokes Equation in the PAX Computer, Proc. 1985 International Conference on Parallel Processing, *IEEE Computer Society* (1985), 426–433.

6. HOSHINO, T., Highly Parallel Computer PAX for Scientific Applications, in "Numerical Methods in Fluid Mechanics II" edited by K. Oshima, Proc. International Symposium on Computational Fluid Dynamics-Tokyo (Japan Society of Computational Fluid Dynamics, Tokyo, 1985), 183–194.

7. LAMBIOTTE, Jr. J. J. and VOIGT R., The Solution of Linear Systems on the CDC STAR-100 Computer, *ACM Trans. Math. Software*, **1** (1975), 308–329.

8. STUBEN, K. and TROTTENBERG, U., Multigrid Methods: Fundamental algorithms, model problem analysis and applications, Proc. 1981 Conference on multigrid Method, Lecture Notes in Mathematics 960, Springer, Berlin, (1982).

(Received November 2, 1987; revised April 18, 1988)