On Implementing An Efficient Iterative Elliptic Solver On A Distributed Parallel System

[1]**Norhashidah Hj. Mohd. Ali,** [2]**Rosni Abdullah,** [3]**Kok Jun Lee,** [4]**Kong Chin Hua**
[1]School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia; e-mail: shidah@cs.usm.my
[2,3,4]School of Computer Science, Universiti Sains Malaysia, 11800 Penang, Malaysia; e-mail: rosni@cs.usm.my

Abstract. In Abdullah (1991), the use of finite difference discretization derived from the rotated (cross) five-point formula [4] in approximating the elliptic equation resulted in a new improved group explicit algorithm, the Explicit Decoupled Group (EDG) method, where the new algorithm was found to be more superior than the explicit methods due to Yousif and Evans (1986). This new iterative algorithm has been developed to be run on the *Sequent Balance*, a shared memory parallel computer ([2], [3], [11]) where it was shown to be viable to be implemented on this type of platform. In this paper, we describe our effort in parallelising this method in solving the two dimensional Poisson equation on a newer parallel paradigm; the distributed parallel system. Communication issues between processors have to be taken into account since data or information are transferred between processors via message passing. Several strategies are proposed on parallelising this method on a cluster of *Sun* workstations; the results of some computational experiments will be reported.

Keywords: Parallel computing; Poisson equation; *rotated* finite difference; Explicit Decoupled Group (EDG) method; Parallel Virtual Machine (PVM)

1. INTRODUCTION

A new concept of explicit methods, pointwise or groupwise, was extensively researched during the mid 1980's as they possess qualities which makes them amenable for use on parallel computers. Such attempts include Evans and Abdullah(1983), and Yousif and Evans(1986), who developed the group explicit methods for the solution of parabolic and elliptic p.d.e.'s respectively. The method was further developed as the Alternating Group Explicit (AGE) method (Evans & Sahimi, 1988), which is an analogue to the famous Alternating Direction Implicit(ADI) method but has the advantage of being explicit and thus very easy to parallelise. Since then, the emergence of newer explicit methods with promising and improved results was greatly observed. Among them are the works of Abdullah (1991) and Yousif & Evans (1995) who developed the four-point Explicit Decoupled Group (EDG) and six to nine-point EDG method respectively in solving the elliptic equation by discretising the p.d.e.'s on rotated grids. In this paper, we study the parallel implementation of the four-point EDG method in solving the two dimensional steady-state elliptic equation on a message passing architecture as a way to further improve the performance of the method. A brief description of the iterative method studied is presented in Section 2. In Sections 3 and 4, we discuss the strategies used for parallelising the method and Section 5 presents the results of experiments performed.

2. THE EXPLICIT DECOUPLED GROUP (EDG) ITERATIVE METHOD

Consider the finite difference discretization schemes for solving the Poisson equation

$$u_{xx} + u_{yy} = f(x,y), \qquad (x,y) \in \Omega, \tag{2.1}$$

with Dirichlet boundary conditions $u(x,y) = g(x,y), \quad (x,y) \in \partial\Omega$. Here Ω is a continuous unit square solution domain. This problem typically arises in the electrostatic, heat conduction, and fluid mechanics areas, among others. Let Ω be discretized uniformly in both x and y directions with a mesh size *h = 1/n*, where *n* is an integer. In solving this problem using the finite difference method, Equation (2.1) may be approximated at the point (x_i, y_j) in many ways. The simplest approximation will result in the following:

$$u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j} - 4u_{ij} = h^2 f_{ij} \; . \tag{2.2}$$

Here, $u_{ij} = u(x_i, y_j)$. This formula is so commonly used that it is known as the five-point difference appproximation. Another difference approximation is (Dahlquist & Bjorck, 1974)

$$u_{i+1,j+1} + u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} - 4u_{ij} = 2h^2 f_{ij} \, . \tag{2.3}$$

This approximation is used in the derivation of the four-point EDG iterative method (Abdullah 1991; Yousif & Evans 1995). Let us now assume that the solution at any four points of the solution domain can be solved using the *rotated* five-point finite difference approximation (2.3). This will result in a (4x4) system of equations

$$\begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & 0 & 0 \\ 0 & 0 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_{ij} \\ u_{i+1,j+1} \\ u_{i+1,j} \\ u_{i,j+1} \end{bmatrix} = \begin{bmatrix} u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} - 2h^2 f_{ij} \\ u_{i,j+2} + u_{i+2,j} + u_{i+2,j+2} - 2h^2 f_{i+1,j+1} \\ u_{i,j-1} + u_{i+2,j-1} + u_{i+2,j+1} - 2h^2 f_{i+1,j} \\ u_{i-1,j+2} + u_{i-1,j} + u_{i+1,j+2} - 2h^2 f_{i,j+1} \end{bmatrix} \tag{2.4}$$

which can be written in a *decoupled* system of (2x2) equations whose explicit forms are given by

$$\begin{bmatrix} u_{ij} \\ u_{i+1,j+1} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} - 2h^2 f_{ij} \\ u_{i,j+2} + u_{i+2,j} + u_{i+2,j+2} - 2h^2 f_{i+1,j+1} \end{bmatrix} \tag{2.5}$$

and

$$\begin{bmatrix} u_{i+1,j} \\ u_{i,j+1} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j-1} + u_{i+2,j-1} + u_{i-2,j+1} - 2h^2 f_{i+1,j} \\ u_{i-1,j+2} + u_{i+1,j} + u_{i+1,j+2} - 2h^2 f_{i,j+1} \end{bmatrix} \tag{2.6}$$

It may be observed that the above two equations and their corresponding computational molecule (Figure 1) that the evaluation of (2.5) involves points of type ○ only, while (2.6) can be evaluated involving points of type □ only. Thus, the calculations of (2.5) and (2.6) can be carried out independently which may save the execution time by nearly half if the iteration is carried out on

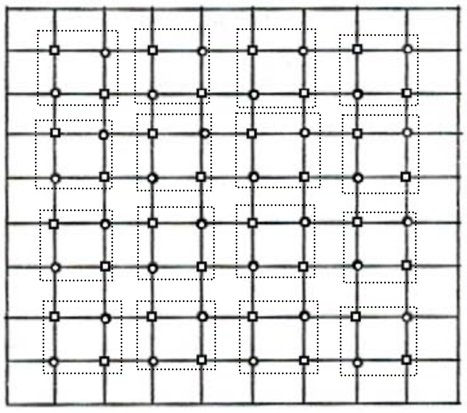


Figure 1. Groups of four points for EDG

only *one* type of points; either the type ○ or □. After convergence is achieved, the solution at the other remaining half of the points will be evaluated directly once using the standard five-point formula (2.2).

3. PARALLELISING STRATEGIES

Strategy 1

In this strategy, the rectangular domain is decomposed into a number of horizontal strips consisting of two rows arranged in the order shown in Figure 2 for the case n = 9. Assuming the iteration involves points of type ○ only, the system of equations that represents the evaluation of solutions at these points using formulae (2.5) on these ordered horizontal strips, is as shown in (3.1)-(3.2):

$$\begin{bmatrix} D & 0 & \vdots & V & 0 \\ 0 & D & \vdots & L & V \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ L & V & \vdots & D & 0 \\ 0 & L & \vdots & 0 & D \end{bmatrix} \begin{bmatrix} \underline{u}_{\text{Strip1}} \\ \underline{u}_{\text{Strip2}} \\ \cdots \\ \underline{u}_{\text{Strip3}} \\ \underline{u}_{\text{Strip4}} \end{bmatrix} = \begin{bmatrix} \underline{b}_{\text{Strip1}} \\ \underline{b}_{\text{Strip2}} \\ \cdots \\ \underline{b}_{\text{Strip3}} \\ \underline{b}_{\text{Strip4}} \end{bmatrix} \tag{3.1}$$

where

$$D = \begin{bmatrix} R_0 & R_1 & & \\ R_1^T & R_0 & R_1 & \\ & R_1^T & R_0 & R_1 \\ & & R_1^T & R_0 \end{bmatrix}, \; L = \begin{bmatrix} R_1 & R_1 & & \\ & R_1 & R_1 & \\ & & R_1 & R_1 \\ & & & R_1 \end{bmatrix}, \; V = L^T$$

with

$$R_0 = \begin{bmatrix} 4 & -1 \\ -1 & 4 \end{bmatrix} \text{and } R_1 = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}. \tag{3.2}$$

The system (3.1) can be rewritten as

$$\begin{bmatrix} \underline{D} & \vdots & C \\ \cdots & \cdots & \cdots \\ F & \vdots & \underline{D} \end{bmatrix} \begin{bmatrix} \underline{u}_{\text{First}} \\ \cdots \\ \underline{u}_{\text{Second}} \end{bmatrix} = \begin{bmatrix} B_{\text{First}} \\ \cdots \\ B_{\text{Second}} \end{bmatrix}. \tag{3.3}$$

Applying the Gauss-Seidel iteration to (3.3), we have

$$\begin{bmatrix} \underline{D} & \vdots & 0 \\ \cdots & \cdots & \cdots \\ F & \vdots & \underline{D} \end{bmatrix} \begin{bmatrix} \underline{u}_{\text{First}} \\ \cdots \\ \underline{u}_{\text{Second}} \end{bmatrix}^{(k+1)} = \begin{bmatrix} B_{\text{First}} \\ \cdots \\ B_{\text{Second}} \end{bmatrix} - \begin{bmatrix} 0 & \vdots & C \\ \cdots & \cdots & \cdots \\ 0 & \vdots & 0 \end{bmatrix} \begin{bmatrix} \underline{u}_{\text{First}} \\ \cdots \\ \underline{u}_{\text{Second}} \end{bmatrix}^{(k)}, \tag{3.4}$$

so that

$$\underline{u}_{\text{First}}^{(k+1)} = \underline{D}^{-1}\left[B_{\text{First}} - C\underline{u}_{\text{Second}}^{(k)}\right], \tag{3.5}$$

followed by

$$\underline{u}_{\text{Second}}^{(k+1)} = \underline{D}^{-1}\left[B_{\text{Second}} - F\underline{u}_{\text{First}}^{(k+1)}\right]. \tag{3.6}$$

If we define the vectors $\underline{u}_{\text{First}}$ as the values of u being updated in the *first* stage (which belong to Strips 1 and 2 in Figure 2), while $\underline{u}_{\text{Second}}$ as the values of u in the *second* stage (in Strips 3 and 4), we can divide an iteration into two stages. From (3.5)-(3.6), it is clear that the evaluations of u in each stage are independent and can be done in parallel. Each strip of two rows is assigned to a processor at a time. This means there are (n-1)/2 strips of two rows to be distributed to the processors. Each processor iterates on its own group of points and checks for its own local convergence. After local convergence is achieved, a check for global convergence is made. The evaluations of solutions at the remaining points (of type □) will be made only after global convergence is achieved by assigning each strip of two rows to a processor in natural order.

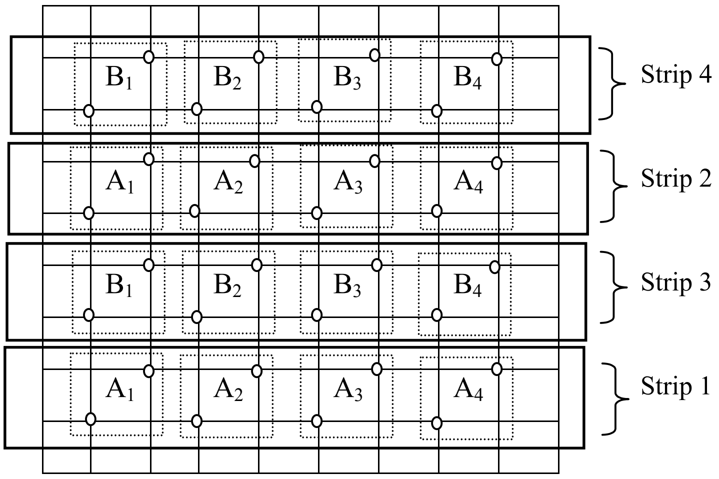


Figure 2. Ordering of group/block of points in Strategy 1

Strategy 2

In this strategy, the four-point groups are assigned to processors in block ordering as shown below for the case n = 9. As illustrated in Figure 3, the iteration are firstly done on the blocks A_1, A_2, A_3, A_4 in parallel, followed by B_1, B_2, B_3, B_4, then C_1, C_2, C_3, C_4 and finally D_1, D_2, D_3, D_4 , all in parallel. By assigning the blocks this way, it may be proven that the computations involved are independent of each other and can be done in parallel.

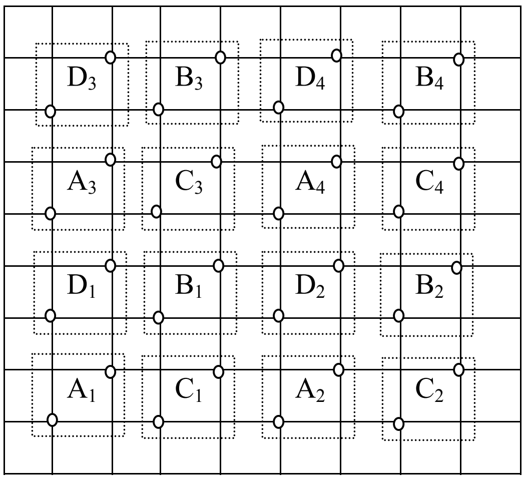


Figure 3. Ordering of group/block of points in Strategy 2

With the group ordering as shown in Figure 3, the system of equations that represents the calculation of solutions at the points ○ is as shown in the system (3.7)-(3.8):

$$\begin{bmatrix} D & C & E & F \\ C^T & D & F^T & E^T \\ E^T & F & D & H \\ F^T & E & H^T & D \end{bmatrix} \begin{bmatrix} \underline{u}_{\text{First}} \\ \underline{u}_{\text{Second}} \\ \underline{u}_{\text{Third}} \\ \underline{u}_{\text{Fourth}} \end{bmatrix} = \begin{bmatrix} B_{\text{First}} \\ B_{\text{Second}} \\ B_{\text{Third}} \\ B_{\text{Fourth}} \end{bmatrix} \tag{3.7}$$

where $$D = \begin{bmatrix} R_0 & & & \\ & R_0 & & \\ & & R_0 & \\ & & & R_0 \end{bmatrix},\ C = \begin{bmatrix} & & & \\ R_1 & & & \\ R_1 & & & \\ & R_1 & R_1 & \end{bmatrix},\ E = \begin{bmatrix} R_1 & & & \\ R_1^T & R_1 & & \\ & & R_1 & \\ & & R_1^T & R_1 \end{bmatrix},$$

$$F = \begin{bmatrix} R_1 & & & \\ & R_1 & & \\ R_1 & & R_1 & \\ & R_1 & & R_1 \end{bmatrix} \text{and } H = \begin{bmatrix} R_1 & & & \\ & R_1 & & \\ & R_1 & R_1 & \\ & & & R_1 \end{bmatrix}, \tag{3.8}$$

with R_0 and R_1 as defined in (3.2). Define the vectors $\underline{u}_{\text{First}} = \left(\underline{u}_{A_1}, \underline{u}_{A_2}, \underline{u}_{A_3}, \underline{u}_{A_4}\right)^T$, $\underline{u}_{\text{Second}} = \left(\underline{u}_{B_1}, \underline{u}_{B_2}, \underline{u}_{B_3}, \underline{u}_{B_4}\right)^T$, $\underline{u}_{\text{Third}} = \left(\underline{u}_{C_1}, \underline{u}_{C_2}, \underline{u}_{C_3}, \underline{u}_{C_4}\right)^T$, $\underline{u}_{\text{Fourth}} = \left(\underline{u}_{D_1}, \underline{u}_{D_2}, \underline{u}_{D_3}, \underline{u}_{D_4}\right)^T$ are the values of u in the first, second, third and fourth stages respectively. Here, $\underline{u}_{A_i}$ $\underline{u}_{B_i}$, $\underline{u}_{C_i}$ and $\underline{u}_{D_i}$ are the *coupled* values of u contained in each block A_i, B_i, C_i and D_i respectively as depicted in Figure 3. The system (3.7) can be solved via Gauss-Seidel iteration as

$$\begin{aligned}
\underline{u}_{\text{First}}^{(m+1)} &= D^{-1}\left[B_{\text{First}} - C\underline{u}_{\text{Second}}^{(m)} - E\underline{u}_{\text{Third}}^{(m)} - F\underline{u}_{\text{Fourth}}^{(m)}\right] \\
\underline{u}_{\text{Second}}^{(m+1)} &= D^{-1}\left[B_{\text{Second}} - C^T\underline{u}_{\text{First}}^{(m+1)} - F^T\underline{u}_{\text{Third}}^{(m)} - E^T\underline{u}_{\text{Fourth}}^{(m)}\right] \\
\underline{u}_{\text{Third}}^{(m+1)} &= D^{-1}\left[B_{\text{Third}} - E^T\underline{u}_{\text{First}}^{(m+1)} - F\underline{u}_{\text{Second}}^{(m+1)} - H\underline{u}_{\text{Fourth}}^{(m)}\right] \\
\underline{u}_{\text{Fourth}}^{(m+1)} &= D^{-1}\left[B_{\text{Fourth}} - F^T\underline{u}_{\text{First}}^{(m+1)} - E\underline{u}_{\text{Second}}^{(m+1)} - H^T\underline{u}_{\text{Third}}^{(m+1)}\right].
\end{aligned} \tag{3.9}$$

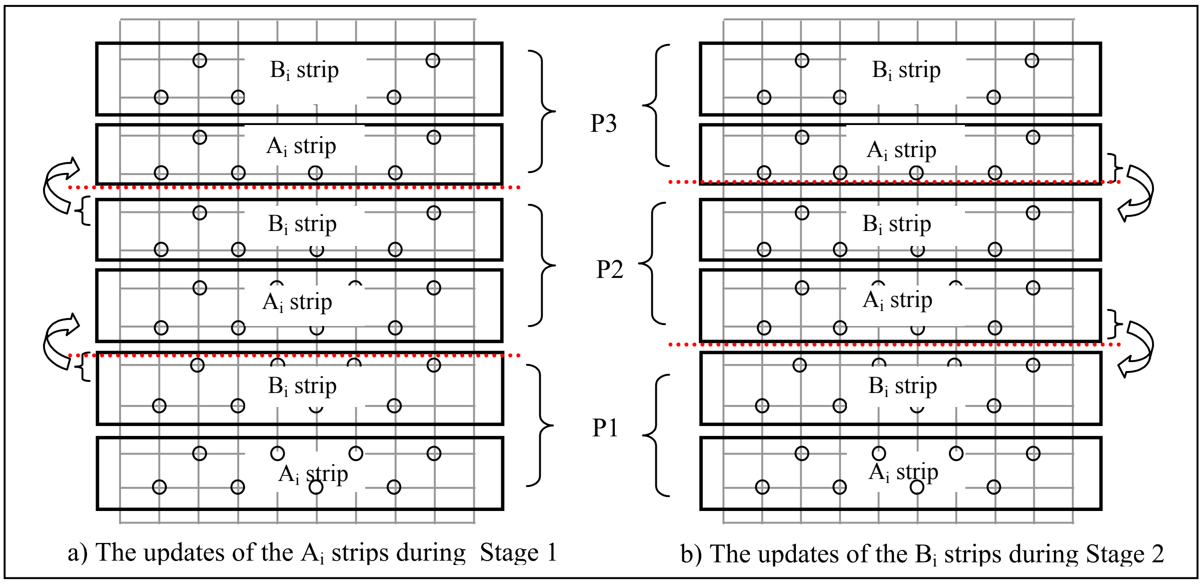
Thus, an iteration is split into four stages where the computations within each stage is independent and can be done in parallel.

4. IMPLEMENTATION ON MESSAGE PASSING SYSTEM

A *parent-child* model consisting of one parent process and one or more than one child processes is used in distributing the tasks [9]. The main responsibility of the parent process is to spawn the child processes, distribute the data to the processes evenly, collect the computed data and print out the results. In addition to these, the parent process will also be involved in the computation processes and will not be left idle. Whilst the child processes will receive the spawned tasks, compute the data, and send the computed data to the parent. Referring to Figures 2-3 the parallel iteration of (3.5)-(3.6) will require that we first update the grid points on the strips 1 and 2, followed by the updates on strips 3 and 4. But the updates on strip 1 (strip 2) require the values at grid points on the strip 3 (strip 4) which may be located in other processors. So these values from other processors must be transmitted to the right processors to achieve the correct results.

4.1 Data Decomposition and Communication Processes for Strategy 1

In this strategy, the horizontal strips of two rows in Figure 2 are distributed evenly to the available processors at each stage. If the nodal point in a particular strip is on the boundary of a subdomain held by a particular processor, then the updated values will have to be exchanged with processors holding adjacent subdomains after each stage in one iteration. In summary there are 2 basic cases which represent the process of distributing the strips to the processors. As an illustration, suppose b is the number of strips for one processor, Figures 4 and 5 illustrate the communications patterns which take place during the updates of points belonging to the A_i and B_i strips respectively when the number of processors is 3. The first case, b = 2k, (k = 1,2,3,4, …) shows the communication patterns needed for the computations involved when the number of strips per process is even. Meanwhile the second diagram illustrates for the case when the number of strips per process is odd (b = 2k+1).



B_i strip
A_i strip
B_i strip
A_i strip
B_i strip
A_i strip
P3
P2
P1
a) The updates of the A_i strips during Stage 1
b) The updates of the B_i strips during Stage 2

Figure 4. Communications patterns between adjacent subdomains in updating the A_i and B_i strips for the case b = 2k (for illustration, here b = 2); communication pattern is the same for all processes

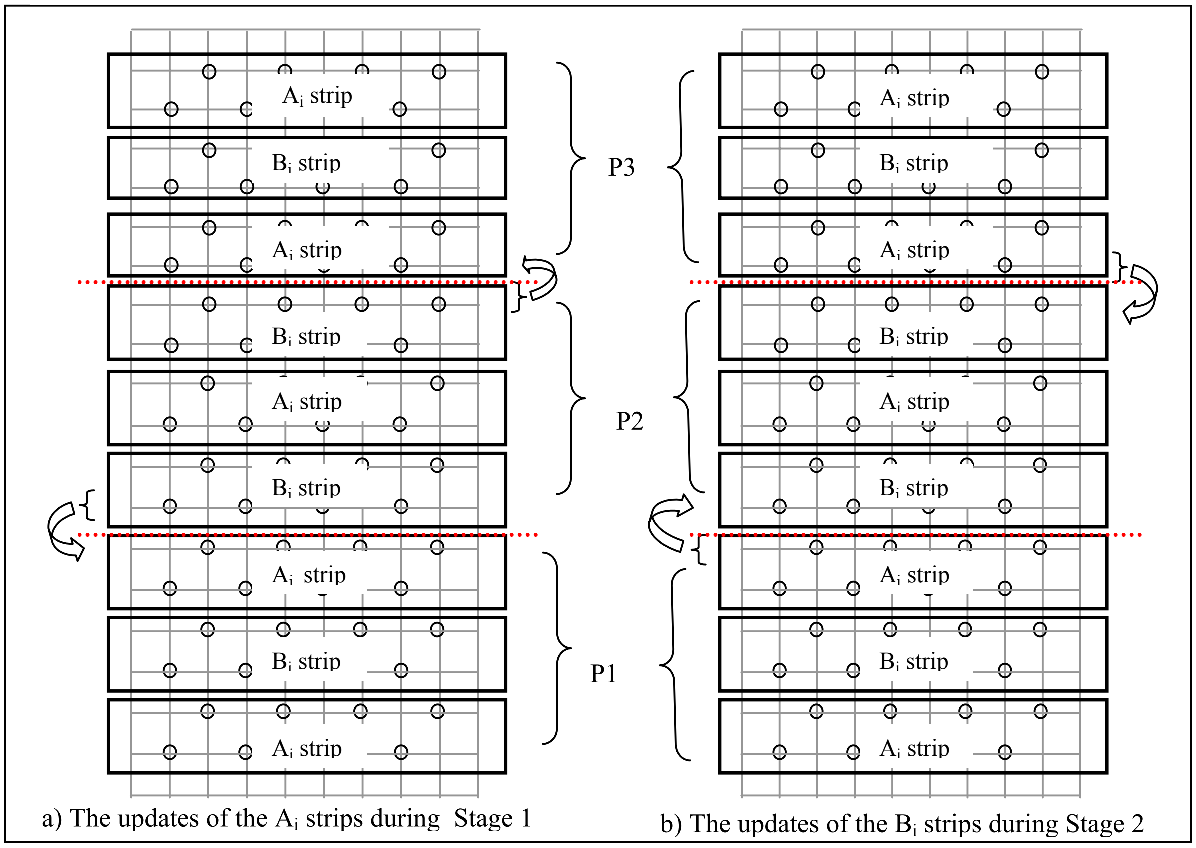


Figure 5. Communications patterns between adjacent subdomains in updating the A_i and B_i strips for the case b = 2k+1 (for illustration, here b = 3); communication pattern is not the same for all processes

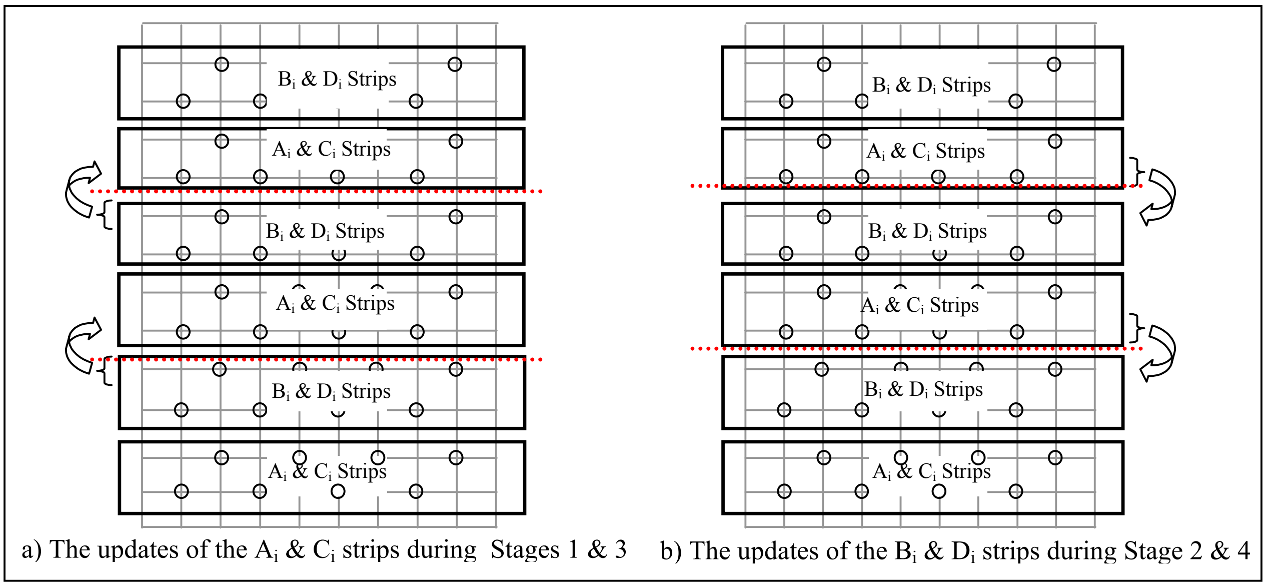

B$_i$ & D$_i$ Strips
A$_i$ & C$_i$ Strips
B$_i$ & D$_i$ Strips
A$_i$ & C$_i$ Strips
B$_i$ & D$_i$ Strips
A$_i$ & C$_i$ Strips
a) The updates of the A$_i$ & C$_i$ strips during Stages 1 & 3
b) The updates of the B$_i$ & D$_i$ strips during Stage 2 & 4

Figure 6. Communications patterns between adjacent subdomains in updating the strips for Strategy 2(b=2k)

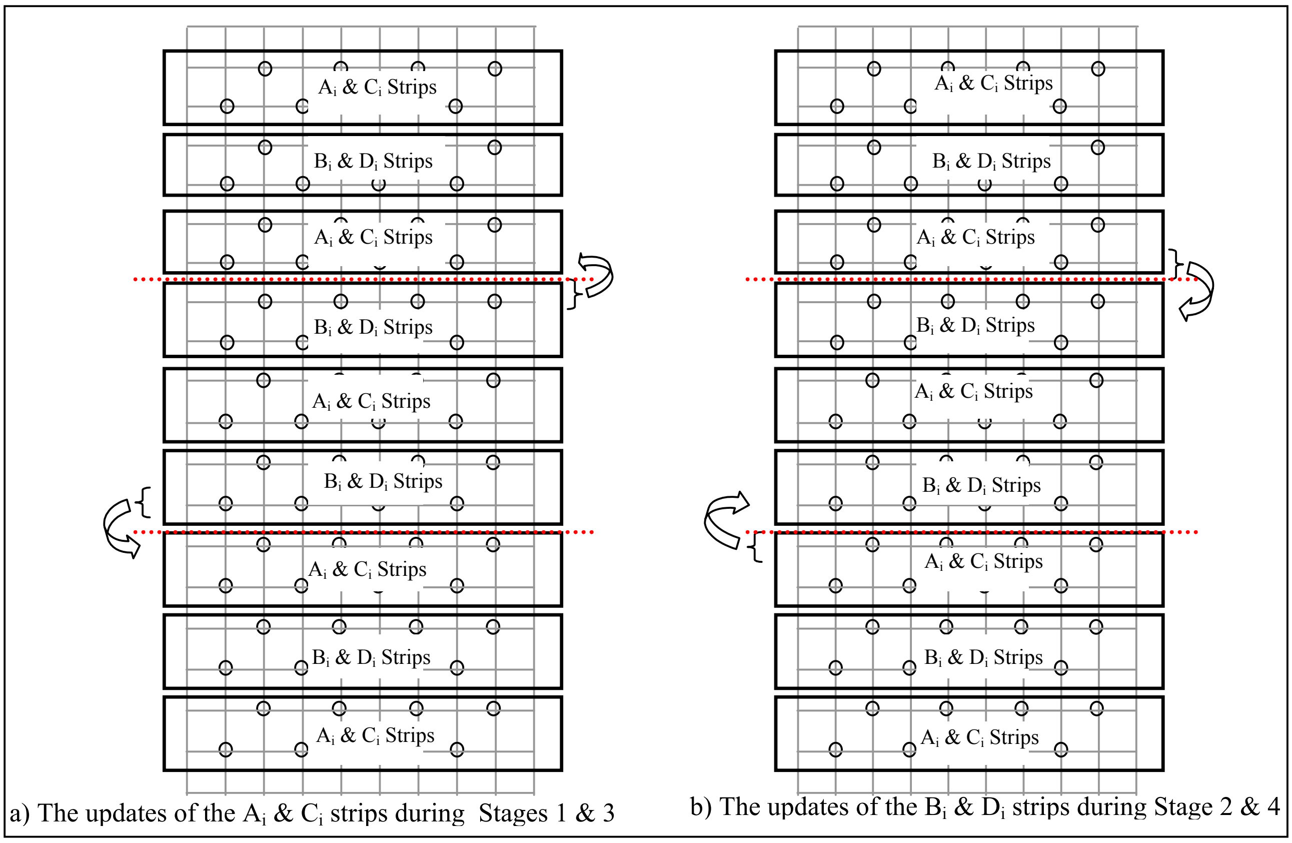


Figure 7. Communications patterns between adjacent subdomains in updating the strips for Strategy 2(b=2k+1)

4.2 Data Composition and Communication Processes for Strategy 2

Similar to Strategy 1, the problem impose different communications patterns amongst the processors in updating the points in the strips when using this strategy. Figures 6(a) and 6(b) illustrates the communication patterns needed in updating the blocks using Strategy 2 when the number of strips per processor is even. Figure 7(a) and 7(b) illustrate the receive-send messages needed for the odd case (b = 2k+1) when updating the perspective blocks.

5. NUMERICAL EXPERIMENTATION

To study the performance of the algorithms, the Poisson equation on the unit square with Dirichlet boundary conditions was used. The model problem chosen was

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = (x^2 + y^2)e^{xy}, \qquad x, y \in \Omega = (0,1)x(0,1) \tag{5.1}$$

with the boundary conditions satisfying its exact solution

$$u(x, y) = e^{xy}.$$

The experiments were ported to run on a cluster of Sun workstations at the School of Computer Science, USM. The *SunOS Release 5.7 Generic* software was used as the operating system supporting the PVM software [7]. In spawning the tasks to the available processors, equal number of strips (where each strip consists of 2 rows) will be distributed to each processor for each strategy. Due to the way the points were grouped, the choice of grid size n is a bit restrictive to ensure that each processor gets equal number of strips at a time i.e. in order to achieve load balancing. For processors equal to 1, 2, 3, 4, 5 and 6 (due to some technical difficulties, larger number of processors were not used in this experiment), we only considered the value of n starting from 121 with the incremental size of 120. Multiple of 120 was chosen because these numbers will ensure that each processor gets equal number of strips whether the number of strip per processor is even or odd. The acceleration parameter ω was chosen to within ± 0.01 that provides the most rapid convergence for each strategy. The local termination test used was the average error test with tolerance $\varepsilon = 1.0x10^{-5}$.

6. RESULTS AND DISCUSSION

Table 1 depicts the CPU-times in seconds, speedup ($S_p = T_1/T_p$, where T_i is the execution time when i number of processors are used), efficiency ($E_p = S_p/p$), error between the exact and computed solution, and the temporal performance (work per second) for EDG method obtained using Strategy 1 and 2. To compare the performance of different algorithms for solving the same problem, the temporal performance metric is used. The temporal performance metric is defined to be the inverse of the execution time T, i.e. $R_T = T^{-1}$ where R_T is solutions per second (sol/s) or timesteps per second (tstep/s) (Hockney 1996). The algorithm with the highest performance executes in the least time and and therefore the better algorithm regardless of the amount of arithmetic involved. P and Effi. represent the number of processors and efficiency values respectively. As depicted in the table, we can see that as the grid sizes increases, the computation gets large which improves the speedup values for EDG. It may also be observed that in Figures 5 and 7, that is when the number of strips per processor is odd, more communication overheads is incurred. This is portrayed in the case n = 601 in Figure 8 where the performance deteriorates slightly when p = 4 in both strategies with Strategy 2 being the worst one among the two. This happens because the value of b is 75, which is odd, which means that each processor gets 75 strips to process. The distribution of tasks will then be of the case b = 2k+1, where the communication pattern is not the same for each processor. There will be processors having to spend 2 units of time; first for sending messages downward and the second to upward adjacent cells in Stage1. Then for Stage 2, they have to spend another 2 units of time; first to receive from the lower adjacent cells, then the second from the upper cells. All together this makes up 4 units of time per iteration. These are the processors which were assigned tasks in the middle part of the grid. At the same time, there are processors which need only spend 1 unit of time to receive messages in Stage 1, and another unit of time to send messages in Stage 2 which makes up 2 units of times per iteration. As a result, there will be processors which remain idle for a few seconds which dampens the performance level. This phenomena is worst in Strategy 2 because of the four stages involved in this strategy. But in all other cases tested, the value of b is even such that the idle time problem is not that significant.

Table 1: Performance of both parallel strategies of EDG

Strategy 1								Strategy 2						
Grid size	P	Times(sec)	Error	Iter	Speed up	Effi.	Work/ secs	Grid size	Times(sec)	Error	Iter	Speed up	Effi.	Work/ secs
n=121	1	1.378989	0.000057	143	1.000	1.000	0.72517	n=121	1.465820	0.000064	146	1.000	1.000	0.68221
w :	2	1.242990	0.000057	143	1.109	0.555	0.80451	w :	1.512552	0.000064	146	0.969	0.485	0.66113
1.9201	3	1.175001	0.000057	143	1.174	0.391	0.85106	1.9210	1.198028	0.000064	146	1.224	0.408	0.83471
	4	1.218844	0.000057	143	1.131	0.283	0.82045		1.595992	0.000064	146	0.918	0.230	0.62657
	5	1.369049	0.000057	143	1.007	0.201	0.73043		1.787144	0.000064	146	0.820	0.164	0.55955
	6	1.704507	0.000057	143	0.809	0.135	0.58668		2.201464	0.000064	146	0.666	0.111	0.45424
n=241	1	10.621968	0.000086	277	1.000	1.000	0.09414	n=241	11.455046	0.000075	284	1.000	1.000	0.08730
w :	2	6.272446	0.000086	277	1.693	0.847	0.15943	w :	7.217392	0.000075	284	1.587	0.794	0.13855
1.9581	3	5.027715	0.000086	277	2.113	0.704	0.19890	1.9583	6.046659	0.000075	284	1.894	0.631	0.16538
	4	4.317121	0.000086	277	2.460	0.615	0.23164		5.232187	0.000075	284	2.189	0.547	0.19112
	5	4.172730	0.000086	277	2.546	0.509	0.23965		5.523861	0.000075	284	2.074	0.415	0.18103
	6	4.501096	0.000086	277	2.360	0.393	0.22217		7.487307	0.000075	284	1.530	0.255	0.13356
n=361	1	35.550673	0.000121	410	1.000	1.000	0.02813	n=361	38.329987	0.00006	421	1.000	1.000	0.02609
w :	2	19.679913	0.000121	410	1.806	0.903	0.05081	w :	22.161947	0.00006	421	1.730	0.865	0.04512
1.9712	3	15.145524	0.000121	410	2.347	0.782	0.06603	1.9712	16.672556	0.00006	421	2.299	0.766	0.05998
	4	12.666251	0.000121	410	2.807	0.702	0.07895		15.928657	0.00006	421	2.406	0.602	0.06278
	5	11.146566	0.000121	410	3.189	0.638	0.08971		13.614610	0.00006	421	2.815	0.563	0.07345
	6	10.692294	0.000121	410	3.325	0.554	0.09353		16.373404	0.00006	421	2.341	0.390	0.06107
n=481	1	83.152665	0.000254	536	1.000	1.000	0.01203	n=481	89.756201	0.000204	550	1.000	1.000	0.01114
w :	2	44.505460	0.000254	536	1.868	0.934	0.02247	w :	48.895314	0.000204	550	1.836	0.918	0.02045
1.9782	3	32.208955	0.000254	536	2.582	0.861	0.03105	1.9781	35.180070	0.000204	550	2.551	0.850	0.02843
	4	25.922891	0.000254	536	3.208	0.802	0.03858		29.901118	0.000204	550	3.002	0.750	0.03344
	5	22.698175	0.000254	536	3.663	0.733	0.04406		27.705918	0.000204	550	3.240	0.648	0.03609
	6	21.245177	0.000254	536	3.914	0.652	0.04707		26.420143	0.000204	550	3.397	0.566	0.03785
n=601	1	162.104721	0.000567	667	1.000	1.000	0.00617	n=601	178.810449	0.00058	691	1.000	1.000	0.00559
w :	2	85.287735	0.000567	667	1.901	0.950	0.01173	w :	93.091039	0.00058	691	1.921	0.960	0.01074
1.9823	3	60.549126	0.000567	667	2.677	0.892	0.01652	1.9822	65.894069	0.00058	691	2.714	0.905	0.01518
	4	48.960110	0.000567	667	3.311	0.828	0.02042		58.454436	0.00058	691	3.059	0.765	0.01711
	5	41.021602	0.000567	667	3.952	0.790	0.02438		48.618808	0.00058	691	3.678	0.736	0.02057
	6	37.402440	0.000567	667	4.334	0.722	0.02674		45.790005	0.00058	691	3.905	0.651	0.02184

7. CONCLUSION

In this work, we presented the experimental results illustrating the parallel implementations of the group explicit elliptic solver, EDG, on a cluster of workstations using PVM. Two parallelising strategies were developed in distributing the computations involved to the available processors. Between the two strategies, it is observed that the execution timings for Strategy 1 is less than Strategy 2 in all of the cases tested. This is due to the fact that the latter strategy impose more communication overheads in receiving and sending messages for the four stages involved in this strategy than the overheads incurred in the former strategy. It is also observed that the speedup and efficiency values for Strategy 1 are slightly better than Strategy 2 in almost all of the cases tested which indicates the amount of computations carried out over the total overheads in Strategy 1 is greater than the one in Strategy 2. In general, the two algorithms developed turned out to be relatively efficient and viable to be ported on a distributed system with Strategy 1 being the best one.

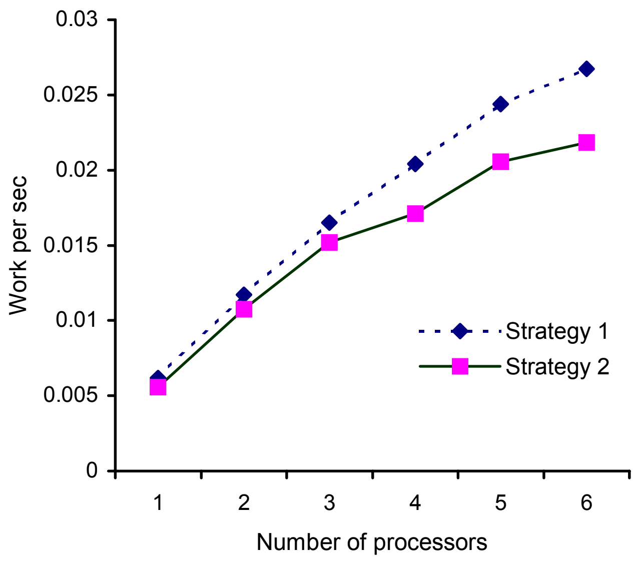


Figure 8 : Temporal performance of EDG when grid size n = 601

Acknowledgement

The authors acknowledge the research grant (304/PMATHS/633054) provided by Universiti Sains Malaysia, Penang that has resulted in this article.

References

1. Abdullah, A.R., 1991, The Four Point Explicit Decoupled Group (EDG) Method : A Fast Poisson Solver, *International Journal of Computer Mathematics.,* **38**, 61-70.
2. Abdullah, A. R. and Ali, N. M., 1996, '*The Comparative Study of Parallel Strategies For The Solution of Elliptic PDE's',* Parallel Algorithms and Applications. **10**: 93-103.
3. Ali, N.H.M. and Abdullah, A.R., 1995, Parallel Four Point Explicit Decoupled Group (EDG) Method For Elliptic PDE's. *Proceedings of the Seventh IASTED/ISMM International Conference on Parallel and Distributed Computing and Systems, Washington D.C.*, pp. 302-304.
4. Dahlquist, G., and Bjorck, A., 1974, *Numerical Methods*. Englewood Cliffs, N.Jersey: Prentice-Hall.
5. Evans, D.J. and Abdullah, A. R., 1983, A New Explicit Method for the Solution of $\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}$, *International Journal of Computer Mathematics*, 14, pp. 325-353.
6. Evans, D.J. and Sahimi, M. S., 1988, The Alternating Group Explicit(AGE) Iterative Method for Solving Parabolic Equations, 1-2 Dimensional Problems, *International Journal of Computer Mathematics*, 24, pp. 250-281.
7. Geist, A. *et al,* 1994, *PVM:Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing*, MIT, Cambridge
8. Hockney, R.W., 1996, *The Science of Computer Benchmarking*, SIAM
9. Wilkinson, B. and Allen, M., 1999, *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall, New Jersey
10. Yousif, W.S. and Evans, D.J., 1986, Explicit group over-relaxation methods for solving elliptic partial differential equations, *Math. Computer Simulation*, **28**, p. 453-466.
11. Yousif, W.S. and Evans, D.J., 1995, Explicit DeCoupled Group Iterative Methods And Their Parallel Implementations, *Parallel Algorithms and Applications* **7**:53-71.