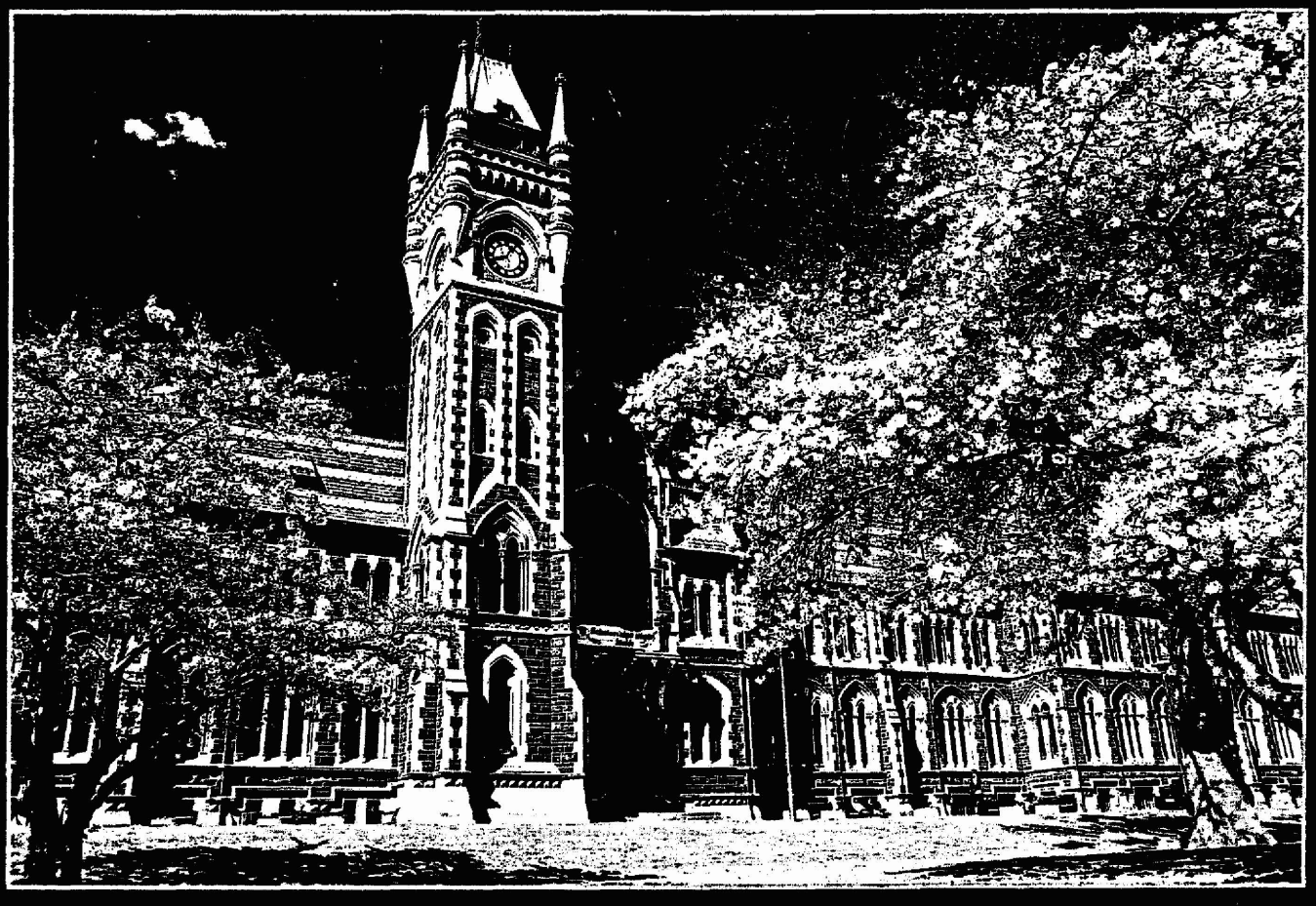
NINTH INTERNATIONAL CONFERENCE ON

# PARALLEL AND DISTRIBUTED COMPUTING, APPLICATIONS AND TECHNOLOGIES

# PDCAT 2008



UNIVERSITY OF OTAGO, DUNEDIN, NEW ZEALAND

1-4 DECEMBER 2008

Editors: Zhiyi Huang, Zhiwei Xu, Nathan Rountree, Laurent Lefevre, Hong Shen, John Hine, and Yi Pan

IEEE computer society

IEEE

UNIVERSITY OF OTAGO
Te Whare Wānanga o Otāgo
NEW ZEALAND

World

ENABLED BY KAREN

*Proceedings*

# Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies

*Dunedin, New Zealand*
*1–4 December, 2008*

**Editors**
Zhiyi Huang, Zhiwei Xu, Nathan Rountree, Laurent Lefevre,
Hong Shen, John Hine, and Yi Pan

**Sponsors**
KAREN — The Kiwi Advanced Research and Education Network
University of Otago
World 45

CPS Conference Publishing Services

Los Alamitos, California

Washington • Tokyo

IEEE computer society

IEEE Computer Society Order Number P3443
BMS Part Number CFP08536-PRT
ISBN 978-0-7695-3443-5
Library of Congress Number 2008934987

*Additional copies may be ordered from*:

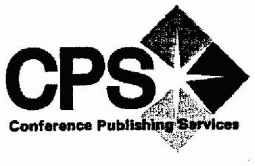
IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

*Individual paper REPRINTS may be ordered at*: <reprints@computer.org>

Editorial production by Lisa O'Conner
Cover art production by Alex Torres
Printed in the United States of America by Applied Digital Imaging


*IEEE Computer Society*
**Conference Publishing Services (CPS)**
http://www.computer.org/cps

# Table of Contents

**PDCAT 2008**

## Parallel/Distributed Algorithms



## Parallel/Distributed Architecture



## Interconnection Networks



## High Performance Computing

# A New Iterative Elliptic PDE Solver On A Distributed PC Cluster

[1]Norhashidah Mohd. Ali and [2]Ng Kok Fu
*School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia*
[1]shidah@cs.usm.my, [2]nkf@myedusoft.com

## Abstract

*Advancement in parallel computers technology has greatly influenced the numerical methods used for solving partial differential equations (pdes). A lot of attention has been devoted to the development of numerical schemes which are suitable for the parallel environment. In this work, we investigate the parallel implementation of the four-point Modified Explicit Decoupled Group (MEDG) method which was introduced by Ali and Ng (2007) as a fast solver for the two dimensional Poisson pde. The method was shown to be more superior than all the methods belonging to the four-points explicit group family namely the Explicit Group (EG) [8], Explicit Decoupled Group (EDG) [1] and Modified Explicit Group (MEG) [7]. This paper presents the preliminary results of the parallel algorithms implemented on a distributed memory PC cluster. Two parallelizing strategies comprising of the two-color zebra and the four-color chessboard orderings in solving a two dimensional Poisson model problem will be discussed.*

## 1. Introduction

Solving partial differential equations (pdes) are usually at the heart of most scientific and engineering applications. The Explicit Decoupled Group (EDG) scheme was developed by Abdullah (1991) as a more efficient Poisson solver on rotated grids by using small fixed size group strategy which was shown to be more economical computationally than the Explicit Group (EG) scheme due to Yousif and Evans [3, 5, 8, 9]. Othman and Abdullah [7] subsequently modified the formulation of the EG method by altering the ordering of grid points taken in the iterative process to come up with the modified four-point EG where this method (MEG) was shown to be more superior in timings than both the original methods. In a recent paper, another explicit group method was proposed, namely the Modified Explicit Decoupled Group (MEDG) method [4] as an addition to this family of four-point explicit group methods in solving Poisson equation. The results obtained indicate that the execution times of MEDG is only about 10% and 15% of those of EG and EDG methods respectively, while MEG is about 14% and 22% of EG and EDG execution times. The MEDG method outperforms MEG in terms of computing time and also exhibits better accuracy in all of the cases observed. The method is explicit in nature so that parallelism is favorable and thus it is worthwhile to investigate its implementation on parallel platforms.

In this paper we present a preliminary study of the newly developed MEDG method on a distributed memory cluster and compare its parallel performance with the other existing explicit group methods. The paper is organized as follows. We present the formulation of the MEDG method in Section 2 followed by the parallel implementation strategies in Section 3. We report the experimental results and concluding remarks in the remaining sections.

## 2. Modified EDG Method

Consider the following two dimensional Poisson equation

$$u_{xx} + u_{yy} = f(x,y), \quad (x,y) \in \Omega, \tag{1}$$

with a Dirichlet boundary condition on a unit square solution domain $[0 \le x, y \le 1]$. Let $\Omega$ be discretized uniformly in both x and y directions with a mesh size $h = 1/n$, where $n$ is a positive even integer. The solutions at the $(n-1)^2$ internal mesh points $(x,y)$ can be approximated by various finite difference schemes. Using the centered difference equation we get the $h$-spaced standard five-point difference formula as follows

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

Rotating the x-y axis clockwise 45° and applying the centered difference formula we get the following rotated five-point difference formula

$$4u_{i,j} - u_{i+1,j+1} - u_{i-1,j-1} - u_{i-1,j+1} - u_{i+1,j-1} = -2h^2 f_{i,j} \tag{3}$$

978-0-7695-3443-5/08 $25.00 © 2008 IEEE
DOI 10.1109/PDCAT.2008.41

Likewise, consider the points at grid size $2h = 2/n$ and apply the centered difference equation will result in the following $2h$-spaced standard five-point difference formula

$$4u_{i,j} - u_{i+2,j} - u_{i-2,j} - u_{i,j+2} - u_{i,j-2} = -4h^2 f_{i,j} \quad (4)$$

and $2h$-spaced rotated five-point difference formula

$$4u_{i,j} - u_{i+2,j+2} - u_{i-2,j-2} - u_{i-2,j+2} - u_{i+2,j-2} = -8h^2 f_{i,j} \quad (5)$$

The MEDG method is based on the above four basic difference formulae.

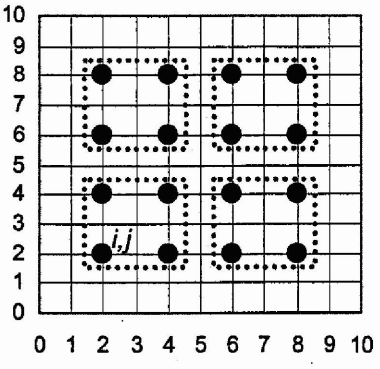



**Figure 1. Groups of four points with $2h$ spacing**

The MEDG method is constructed as follows. We apply Eq. (5) to groups of four points as shown in Figure 1 and produce the following (4x4) system of equations

$$\begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & 0 & 0 \\ 0 & 0 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j} \\ u_{i+2,j+2} \\ u_{i+2,j} \\ u_{i,j+2} \end{bmatrix} = \begin{bmatrix} u_{i-2,j-2} + u_{i+2,j-2} + u_{i-2,j+2} - 8h^2 f_{i,j} \\ u_{i,j+4} + u_{i+4,j} + u_{i+4,j+4} - 8h^2 f_{i+2,j+2} \\ u_{i,j-2} + u_{i+4,j-2} + u_{i+4,j+2} - 8h^2 f_{i+2,j} \\ u_{i-2,j+4} + u_{i-2,j} + u_{i+2,j+4} - 8h^2 f_{i,j+2} \end{bmatrix} \quad (6)$$

which can be inverted and rewritten in explicit forms of a decoupled system of (2x2) equations as follows:

$$\begin{bmatrix} u_{i,j} \\ u_{i+2,j+2} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-2,j-2} + u_{i+2,j-2} + u_{i-2,j+2} - 8h^2 f_{i,j} \\ u_{i,j+4} + u_{i+4,j} + u_{i+4,j+4} - 8h^2 f_{i+2,j+2} \end{bmatrix} \quad (7)$$

$$\begin{bmatrix} u_{i+2,j} \\ u_{i,j+2} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i,j-2} + u_{i+4,j-2} + u_{i-4,j+2} - 8h^2 f_{i+2,j} \\ u_{i-2,j+4} + u_{i+2,j} + u_{i+2,j+4} - 8h^2 f_{i,j+2} \end{bmatrix} \quad (8)$$

Figure 2 and Figure 3 illustrate the computational molecules for Eq. (7) and Eq. (8) respectively. Figure 4 shows the discretization points of a unit square domain with $n$=18 and the various types of points involved. There are $(n-1)^2$=289 internal points that need to be solved numerically. Of these solution points, only 32 points (either of type ● or type ▲) that will be computed iteratively using the MEDG method.

It is obvious that the evaluation of Eq. (7) involves only points of type ● and Eq. (8) ) involves only points of type ▲.

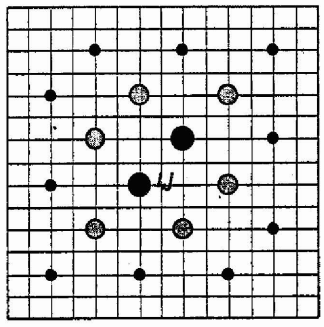



**Figure 2. Computational molecule for Eq.(7)**

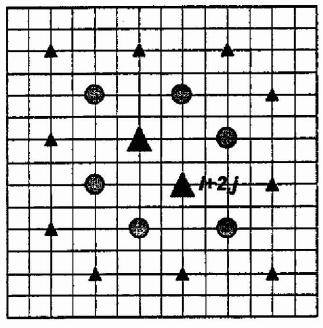



**Figure 3. Computational molecule for Eq.(8)**

We solve the points of type ● iteratively using Eq. (7) until convergence is achieved, after which the points of type ▲ are computed directly once using the $2h$-spaced standard five-point formula of Eq. (4). The remaining in-between points of type □ are also computed directly once using the $h$-spaced rotated five-point difference formula of Eq. (3), and followed by points of type ◇ using the $h$-spaced standard five-point difference formula of Eq. (2).

We now define the four point MEDG method with successive over-relaxation (SOR) iterative scheme as follows.

1. Divide the solution domain into five types of points as shown in Figure 4 (in this case $n$=18).
2. Group all the $2h$-spaced ● and ▲ points into four-point groups.
3. Iterate the intermediate solution of points ● in each group using

$$\begin{bmatrix} \hat{u}_{i,j}^{(k+1)} \\ \hat{u}_{i+2,j+2}^{(k+1)} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} u_{i-2,j-2}^{(k+1)} + u_{i+2,j-2}^{(k+1)} + u_{i-2,j+2}^{(k)} - 8h^2 f_{i,j} \\ u_{i,j+4}^{(k)} + u_{i+4,j}^{(k)} + u_{i+4,j+4}^{(k)} - 8h^2 f_{i+2,j+2} \end{bmatrix}$$

and implement the relaxation scheme

$$\begin{bmatrix} u_{i,j}^{(k+1)} \\ u_{i+2,j+2}^{(k+1)} \end{bmatrix} = \begin{bmatrix} u_{i,j}^{(k)} \\ u_{i+2,j+2}^{(k)} \end{bmatrix} + \omega \left( \begin{bmatrix} \hat{u}_{i,j}^{(k+1)} \\ \hat{u}_{i+2,j+2}^{(k+1)} \end{bmatrix} - \begin{bmatrix} u_{i,j}^{(k)} \\ u_{i+2,j+2}^{(k)} \end{bmatrix} \right)$$

4. If the solution converge, go to step 5. Otherwise, repeat the iteration step 3.
5. Finally evaluate the remaining points in this sequence:
   a. points of type ▲

$$u_{i,j} = \frac{1}{4}\left(u_{i+2,j} + u_{i-2,j} + u_{i,j+2} + u_{i,j-2} - 4h^2 f_{i,j}\right)$$

   b. points of type □

$$u_{i,j} = \frac{1}{4}\left(u_{i+1,j+1} + u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} - 2h^2 f_{i,j}\right)$$

   c. points of type ◇

$$u_{i,j} = \frac{1}{4}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - h^2 f_{i,j}\right)$$

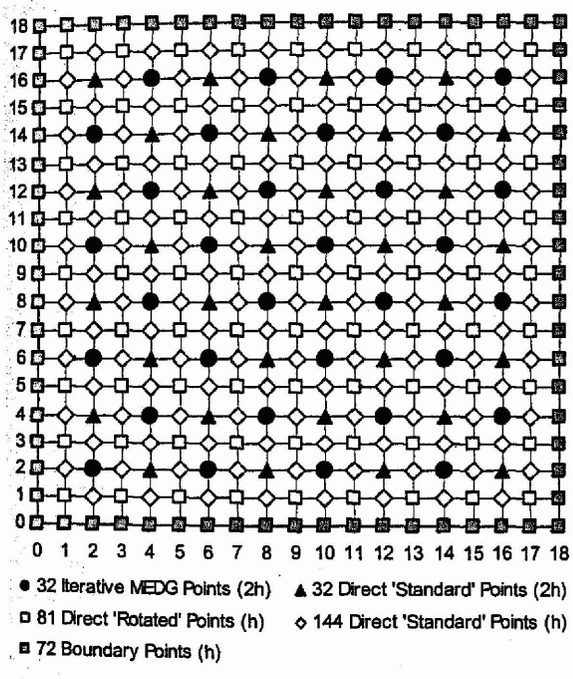



**Figure 4. Types of discretized points in MEDG method for $n$=18**

## 3. Parallel Modified EDG Method

We describe two ordering strategies used in parallelizing the MEDG method, specifically the 2-color-zebra and 4-color-chessboard orderings. We use the lower-left and upper-right pair of points for the iteration process of the MEDG method in conjunction with Eq. (7).

### 3.1. Two-color zebra ordering

In this strategy, the $2h$-spaced MEDG iterative points (see Figure 4) of domain $\Omega$ are marked with two colors A and B resulting in alternating horizontal panels as shown in Figure 5. Each panel is comprised of four rows of points. Suppose that $n$ = 34, then there are $(n$-2)/4=8 panels all together. Using Eq. (7), these 4-row panels are evenly distributed to the number of processors available, $p$, for parallel computation. In our experiments, we chose $n$ such that ($n$-2) mod $4p$=0. Thus each processor has its own local domain of $u$ with equal number of panels which is given by

$$N(n,p) = \frac{(n-2)}{4p} \qquad (9)$$

Note that Eq. (9) is true regardless of the ordering strategies used. This also holds for the MEG method. For the EG and EDG methods, each panel is comprised of two rows of points, making the number of panels each processor receives to be twice the number given in Eq. (9).

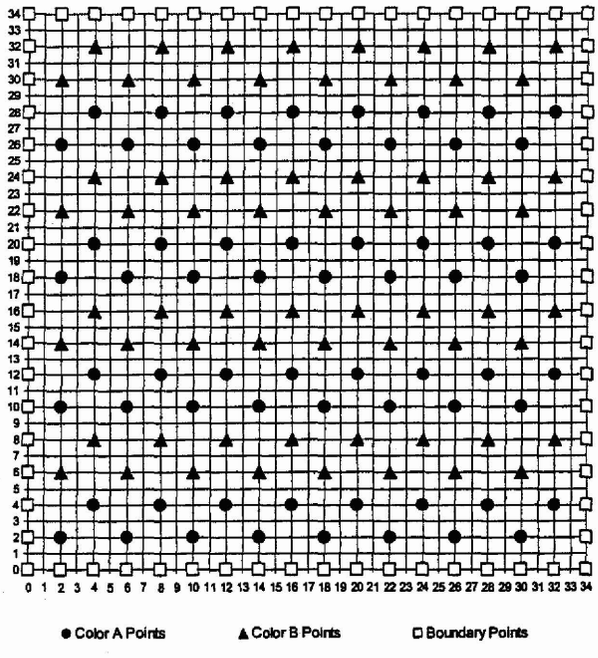



**Figure 5. 2-color zebra ordering for MEDG method**

All the A colored panels are computed first simultaneously by all processors (stage 1), followed by the computation of the B colored panels (stage 2). In each stage, left-right and bottom-up compute sequence is employed. Since the computation of A panels requires the values of points in B panels and vice versa, communication needs to be carried out to send the

values of local domain boundary points to adjacent processors. So a single iteration in each processor will consist of a series of sequential stages: stage 1 where A panels are computed and checked for local convergence followed by data transfer of any A panel boundaries, then stage 2 is performed where B panels are computed and checked for local convergence followed by data transfer of any B panel boundaries. A global convergence test is then performed before the next iteration. The iteration terminates if the global convergence is achieved.

Two possible cases may exist, depending on the number of panels each processor receives, $N(n,p)$ as listed in Table 1. For example, in an experiment with n=482 and 5 processors, each processor receives $N$=480/20=24 panels which results in an arrangement with A panel as lower boundary and B panel as upper boundary for all processors (case $N$%2=0). In another experiment with the same $n$ but 8 processors, each processor receives $N$=480/32=15 panels (case $N$%2=1) which results in an arrangement with A panel as both lower and upper boundary for processors 1, 3, 5 and 7 ($pID$%2=1), and a different arrangement with B panel as both lower and upper boundary for processors 2, 4, 6 and 8 ($pID$%2=0). From the communication pattern shown in Table 1, it is expected that the latter case will incur more overheads resulting in higher execution time than the former case.

**Table 1. Two possible cases for 2-color zebra ordering**

**(a) Case $N$%2 = 0**

| Processor | Panel Arrangement | Computation & communication tasks in an iteration loop Stage 1 | Stage 2 |
|---|---|---|---|
| All | BB...<br>AA... | Compute A<br>Send A below, Recv A above | Compute B<br>Send B above, Recv B below |

**(b) Case $N$%2 = 1**

| Processor | Panel Arrangement | Computation & communication tasks in an iteration loop Stage 1 | Stage 2 |
|---|---|---|---|
| $pID$%2=0 | BB...<br>AA...<br>BB... | Compute A<br>Recv A above, Recv A below | Compute B<br>Send B below, Send B above |
| $pID$%2=1 | AA...<br>BB...<br>AA... | Compute A<br>Send A below, Send A above | Compute B<br>Recv B above, Recv B below |

## 3.2. Four-color chessboard ordering

In this strategy, four colors are used to mark the $2h$-spaced MEDG iterative points in the way much like the color arrangement in a chessboard but with four colors (Figure 6). So the iteration loop will consist of four stages of computation and communication. This incur extensive communication compared to the 2-color zebra ordering. However, an appropriate combination of $n$ and $p$ values will reduce some communication costs.

**Table 2. Two possible cases for 4-color chessboard ordering**

**(a) Case $N$%2 = 0**

| Processor | Panel Arrangement | Computation & communication tasks in an iteration loop Stage 1 | Stage 2 | Stage 3 | Stage 4 |
|---|---|---|---|---|---|
| All | CBCB...<br>ADAD... | Com pute A<br>Send A below<br>Recv A above | Com pute B<br>Send B above<br>Recv B below | Com pute C<br>Send C above<br>Recv C below | Com pute D<br>Send D below<br>Recv D above |

**(b) Case $N$%2 = 1**

| Processor | Panel Arrangement | Computation & communication tasks in an iteration loop Stage 1 | Stage 2 | Stage 3 | Stage 4 |
|---|---|---|---|---|---|
| $pID$%2 =0 | CBCB...<br>ADAD...<br>CBCB... | Com pute A<br>Recv A below<br>Recv A above | Com pute B<br>Send B above<br>Send B below | Com pute C<br>Send C above<br>Send C below | Com pute D<br>Recv D below<br>Recv D above |
| $pID$%2 =1 | ADAD...<br>CBCB...<br>ADAD... | Com pute A<br>Send A above<br>Send A below | Com pute B<br>Recv B below<br>Recv B above | Com pute C<br>Recv C below<br>Recv C above | Com pute D<br>Send D above<br>Send D below |

With this ordering, two possible cases may exist depending on the panel arrangement at the boundaries, as shown in Table 2. In the first case where $N(n,p)$%2=0, each processor incurs exactly one send and one receive communication in each stage. In the second case where $N(n,p)$%2=1, all processors with $pID$%2=1 will incur two sends in stage 1 and stage 4, while processors with $pID$%2=0 will incur two sends in stage 2 and stage 3. Since two sends executed by the same processors always incur more overheads than one single send, the second case should be avoided.

This particular sequence of computation in 4-color chessboard ordering also allow some degree of overlapping computation with communication for optimizing parallel performance of the MEDG method. Note that the computation of C panels in stage 3 does not require any values of B panels from adjacent

processors, so it can be overlapped with the communication of B boundaries of stage 2. Similarly, the computation of A panels in stage 1 can be overlapped with the communication of D boundaries of stage 4 from the previous iteration. On the other hand, overlapping the computations of B or D panels with communications of A or C panels is not possible since they require the newly updated values from the previous stages (see Figure 2). However a full overlapping of computation-communication is possible for all stages in the MEG method due to its different computational molecule.

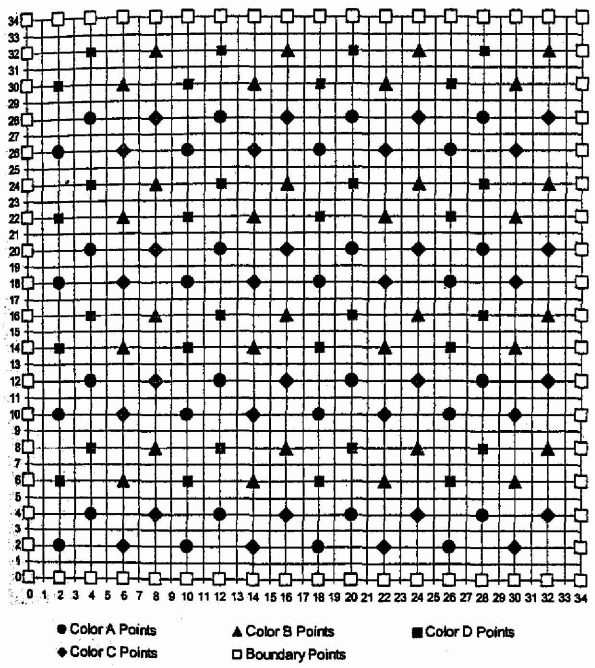



**Figure 6. 4-color chessboard ordering for the MEDG method**

## 4. Experimental Results and Discussion

The following model problem was used as the test problem: $\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=(x^2+y^2)e^{xy},\ x,y\in\Omega$ (10)

with Dirichlet boundary conditions satisfying its exact solution $u(x,y)=e^{xy}$, $(x,y)\in\partial\Omega$, $\partial\Omega$ is the boundary of $\Omega$. We carried out all our experiments on Aurora cluster of Grid Computing Lab at the School of Computer Science, Universiti Sains Malaysia. The master host has two Pentium III 1400MHz CPUs with 3.90 GB of RAM and 143.763 GB of local disk. The 14 worker hosts have each two similar 1400MHz CPUs with 0.99 GB of RAM and 77.739 GB of local disk. All PCs are connected back to back via Gigabit Ethernet NICs, with Linux kernel 2.6.9-42.0.2.ELsmp (x86) and MPI implementation of LAM 7.1.1/MPI 2 C++/ROMIO.

The MEDG method as well as the other three explicit group methods were run with increasing problem sizes $n$=482, 962, 1442, 1922 and 2402, using number of processors $p$=1, 2, 3, 4, 5, 8, 10, 12, and 15. These numbers of processor were chosen so that a manageable range of problem sizes can be used. All combinations of these $n$ and $p$ values, except $n$=482, 1442, 2402 for $p$=8, produce the optimal cases ($N$%2=0) for the 2-color zebra and 4-color chessboard orderings. The experimental values of relaxation factor $\omega_e$ for all the methods were obtained to within ±0.0001 by choosing the ones that resulted in the least number of iterations. In all experiments, we employed $l_\infty$ norm with tolerance, $\varepsilon\leq 10^{-5}$ as the convergence criteria.

Table 3 shows some performance comparison between MEDG and the other explicit group methods for the 2-color zebra ordering. Only parallel execution time (in secs) for 15 processors is shown in the table. It is observed from the results that the MEDG method is faster than the other methods in terms of execution speed. It is due to the fact that its number of iterative points (see Figure 4) is only $(m\text{-}1)^2/8$ compared to $m^2$, $(m^2+1)/2$, and $(m\text{-}1)^2/4$ for EG, EDG and MEG methods respectively [4], where $m$=$n$-1. MEDG also exhibits better accuracy in almost all of the cases observed.

**Table 3. MEDG compared with EG, EDG and MEG methods for 2-color zebra ordering, with parallel execution using 15 processors**

| Method | $N(n,15)$ | Opt w | Serial Time | Par Time | S'Up | Iter # | Max Err | Ave Err |
|---|---|---|---|---|---|---|---|---|
| n = 482 | | | | | | | | |
| EG | 16 | 1.9809 | 10.35 | 1.68 | 6.15 | 698 | 0.536e-3 | 0.201e-3 |
| EDG | 16 | 1.9790 | 8.05 | 1.03 | 7.78 | 580 | 0.385e-3 | 0.146e-3 |
| MEG | 8 | 1.9631 | 2.32 | 0.55 | 4.20 | 357 | 0.226e-3 | 0.081e-3 |
| **MEDG** | **8** | **1.9584** | **1.97** | **0.40** | **4.93** | **305** | **0.236e-3** | **0.090e-3** |
| n = 962 | | | | | | | | |
| EG | 32 | 1.9900 | 78.41 | 8.55 | 9.17 | 1347 | 1.466e-3 | 0.565e-3 |
| EDG | 32 | 1.9891 | 63.79 | 5.97 | 10.68 | 1123 | 0.934e-3 | 0.361e-3 |
| MEG | 16 | 1.9808 | 20.83 | 2.36 | 8.82 | 697 | 0.571e-3 | 0.214e-3 |
| **MEDG** | **16** | **1.9784** | **17.40** | **1.72** | **10.10** | **592** | **0.545e-3** | **0.210e-3** |
| n = 1442 | | | | | | | | |
| EG | 48 | 1.9933 | 252.08 | 23.72 | 10.63 | 1939 | 2.115e-3 | 0.813e-3 |
| EDG | 48 | 1.9926 | 220.49 | 18.27 | 12.07 | 1655 | 1.392e-3 | 0.542e-3 |
| MEG | 24 | 1.9869 | 68.59 | 6.85 | 10.02 | 1025 | 0.978e-3 | 0.374e-3 |
| **MEDG** | **24** | **1.9853** | **58.62** | **5.40** | **10.85** | **873** | **0.850e-3** | **0.329e-3** |
| n = 1922 | | | | | | | | |
| EG | 64 | 1.9949 | 572.03 | 51.54 | 11.10 | 2534 | 2.963e-3 | 1.143e-3 |
| EDG | 64 | 1.9943 | 493.45 | 40.25 | 12.26 | 2158 | 2.373e-3 | 0.929e-3 |
| MEG | 32 | 1.9900 | 158.79 | 14.68 | 10.82 | 1345 | 1.448e-3 | 0.557e-3 |
| **MEDG** | **32** | **1.9888** | **135.34** | **11.76** | **11.51** | **1148** | **1.187e-3** | **0.461e-3** |
| n = 2402 | | | | | | | | |
| EG | 80 | 1.9959 | 1114.44 | 94.11 | 11.84 | 3115 | 3.526e-3 | 1.362e-3 |
| EDG | 80 | 1.9954 | 969.59 | 74.54 | 13.01 | 2666 | 2.860e-3 | 1.120e-3 |
| MEG | 40 | 1.9919 | 299.48 | 26.83 | 11.16 | 1664 | 1.861e-3 | 0.717e-3 |
| **MEDG** | **40** | **1.9909** | **256.38** | **21.72** | **11.80** | **1416** | **1.624e-3** | **0.633e-3** |

Table 4 shows the performance comparison between MEDG and the other explicit group methods for the 4-color chessboard ordering. Figures 7 and 8 show the speedups of the MEDG method compared with the other explicit group methods for the two parallel strategies. All figures show that the MEDG method is able to scale well with the increasing number of processors. The figures also show that the MEDG method has better speedups compared to the MEG method. This is in line with our findings in [6] that EDG method has higher computation to communication ratio compared with EG method in a fast computing cluster such as the Aurora cluster. Since MEDG and MEG methods are derived from the EDG and EG methods respectively, they exhibit similar performance of the original methods. Note that the performance of MEDG declines slightly compared with MEG for the 4-color chessboard ordering due to its restricted overlapping computation with communication as discussed earlier. In all occasions, both EDG and EG methods show better speedups over MEDG and MEG since they have more iterative points to compute compared to each of their modified counterparts which increase their compute-communication ratios.

As mentioned earlier, the performance decline in non-optimal cases can be observed from Figures 7 to 8. Non-optimal cases of the 2-color zebra and 4-color chessboard orderings, are also detected at $p$=8 in Figure 7 and Figure 8. Note that the performance declines are more obvious in MEG and MEDG methods compared to EG and EDG methods due to their lower computation-communication ratio.

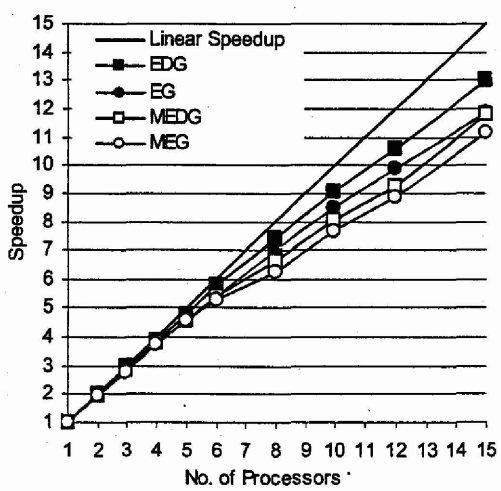



**Figure 7. Speedup at $n$=2402 for 2-color zebra**

**Table 4. MEDG compared with EG, EDG and MEG methods for 4-color chessboard ordering, with parallel execution using 15 processors**

| Method | $N(n,15)$ | Opt w | Serial Time | Par Time | S'Up | Iter # | Max Err | Ave Err |
|---|---|---|---|---|---|---|---|---|
| n = 482 | | | | | | | | |
| EG | 16 | 1.9812 | 18.24 | 1.85 | 9.88 | 649 | 0.463e-3 | 0.186e-3 |
| EDG | 16 | 1.9793 | 16.36 | 1.24 | 13.16 | 607 | 0.266e-3 | 0.107e-3 |
| MEG | 8 | 1.9630 | 4.10 | 0.64 | 6.41 | 335 | 0.277e-3 | 0.113e-3 |
| **MEDG** | **8** | **1.9593** | **3.28** | **0.46** | **7.08** | **313** | **0.168e-3** | **0.067e-3** |
| n = 962 | | | | | | | | |
| EG | 32 | 1.9903 | 142.96 | 11.53 | 12.40 | 1254 | 1.071e-3 | 0.430e-3 |
| EDG | 32 | 1.9892 | 130.36 | 9.46 | 13.77 | 1172 | 0.694e-3 | 0.281e-3 |
| MEG | 16 | 1.9809 | 34.96 | 3.25 | 10.74 | 657 | 0.566e-3 | 0.229e-3 |
| **MEDG** | **16** | **1.9784** | **25.52** | **2.35** | **10.86** | **610** | **0.539e-3** | **0.217e-3** |
| n = 1442 | | | | | | | | |
| EG | 48 | 1.9933 | 473.18 | 37.79 | 12.52 | 1855 | 2.081e-3 | 0.841e-3 |
| EDG | 48 | 1.9926 | 444.44 | 32.42 | 13.71 | 1721 | 1.200e-3 | 0.486e-3 |
| MEG | 24 | 1.9870 | 114.24 | 10.42 | 10.97 | 960 | 1.008e-3 | 0.406e-3 |
| **MEDG** | **24** | **1.9853** | **84.15** | **8.34** | **10.09** | **893** | **0.852e-3** | **0.344e-3** |
| n = 1922 | | | | | | | | |
| EG | 64 | 1.9949 | 1088.77 | 87.19 | 12.49 | 2435 | 2.821e-3 | 1.142e-3 |
| EDG | 64 | 1.9943 | 1014.07 | 75.01 | 13.52 | 2251 | 1.938e-3 | 0.785e-3 |
| MEG | 32 | 1.9901 | 268.43 | 22.37 | 12.00 | 1265 | 1.404e-3 | 0.565e-3 |
| **MEDG** | **32** | **1.9888** | **196.27** | **17.52** | **11.20** | **1175** | **1.149e-3** | **0.465e-3** |
| n = 2402 | | | | | | | | |
| EG | 80 | 1.9959 | 2103.28 | 161.8 | 12.99 | 2971 | 3.544e-3 | 1.428e-3 |
| EDG | 80 | 1.9954 | 1977.99 | 141.7 | 13.96 | 2766 | 2.411e-3 | 0.977e-3 |
| MEG | 40 | 1.9921 | 515.43 | 42.11 | 12.24 | 1579 | 1.328e-3 | 0.535e-3 |
| **MEDG** | **40** | **1.9909** | **375.37** | **31.37** | **11.97** | **1448** | **1.547e-3** | **0.626e-3** |

It can clearly be seen from the figures that the 4-color chessboard ordering is slower than the 2-color zebra ordering. Generally, the 4-color chessboard ordering would produce the most accurate solution almost all the time. However, due its slower execution, it could not be considered as the best candidate for implementing the algorithm in parallel compared to the 2-color ordering strategy.

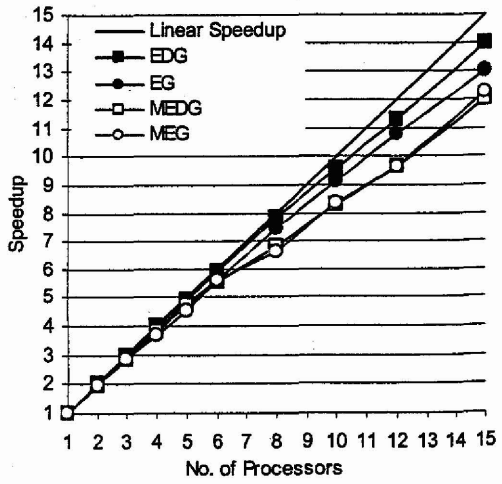



**Figure 8. Speedup at $n$=2402 for 4-color chessboard**

## 5. Conclusions

In this paper, we present a preliminary report on the parallel implementation of the new MEDG method using two ordering strategies running on a PC cluster, and compare them with the existing methods belonging to the four-points explicit group family. The experimental results indicate that the method is the most superior method in terms of execution speed and accuracy. It is also able to scale well with the increasing number of processors and exhibit better speedups compared to the MEG method. Among the two ordering strategies, the 2-color zebra ordering is the best strategy for implementing the MEDG method in parallel. In conclusion, the newly developed MEDG method is able to benefit from parallelism when implemented on the PC cluster.


## Acknowledgement

The authors acknowledge the Fundamental Research Grant (203/PMATHS/671052) for providing the funds to cover for the expenses incurred in attending the PDCAT 2008 conference.


## References


[1] A. R. Abdullah, The Four Point Explicit Decoupled Group (EDG) Method : A Fast Poisson Solver, *International Journal of Comp. Math.*, Vol. 38, 1991, pp. 61-70.

[2] A. R. Abdullah and N. M. Ali, The comparative study of parallel strategies for the solution of elliptic PDE's, *Parallel Algorithms and Applications*. Vol 10, 1996, pp. 93-103.

[3] N. H. M. Ali, A. R. Abdullah and J. L. Kok, A comparative study of explicit group iterative solvers on a cluster of workstations, *Parallel Algorithms and Applications*. Vol 19 (4), 2004, pp. 237-255.

[4] N. H. M. Ali and K. F. Ng, Modified explicit decoupled group method in the solution of 2-D elliptic PDEs. *Proceedings of the 12th WSEAS International Conference on Applied Mathematics*, Cairo, Egypt, 29-31 December 2007, pp. 162-167.

[5] D. J. Evans and W. S. Yousif, The Implementation of The Explicit Block Iterative Methods On the Balance 8000 Parallel Computer, *Parallel Computing*, Vol 16, 1990, pp. 81-97.

[6] K. F. Ng and N. H. M. Ali, Performance analysis of explicit group parallel algorithms for distributed memory multicomputer, *Parallel Computing* (Special Issue: PMAA '06), Vol. 34 (6-8), 2008, pp. 427-440.

[7] M. Othman and A. R. Abdullah, An Efficient Four Points Modified Explicit Group Poisson Solver, *International Journal of Computer Mathematics*, Vol. 76, 2000, pp. 203-217.

[8] W. S. Yousif and D. J. Evans, Explicit Group Over-relaxation Methods for Solving Elliptic Partial Differential Equations, *Mathematics and Computers in Simulation*, Vol. 28, 1986, pp. 453-466.

[9] W. S. Yousif and D. J. Evans, Explicit de-coupled group iterative methods and their parallel implementation, *Parallel Algorithms and Applications*. Vol 7, 1995, pp. 53-71.