

UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN
LAPORAN AKHIR PENYELIDIKAN

TAJUK PROJEK : **Parallel strategies on a distributed parallel computer system**
vot: 75019, Ketua projek: Dr. Norma Alias

Saya **DR. NORMA ALIAS**
(HURUF BESAR)

Mengaku membenarkan **Laporan Akhir Penyelidikan** ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Laporan Akhir Penyelidikan ini adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan rujukan sahaja.
3. Perpustakaan dibenarkan membuat penjualan salinan Laporan Akhir Penyelidikan ini bagi kategori TIDAK TERHAD.
4. * Sila tandakan (/)

☐

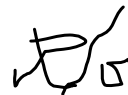
SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau Kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972).

☐

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh Organisasi/badan di mana penyelidikan dijalankan).

☒TIDAK
TERHAD

TANDATANGAN KETUA PENYELIDIK

Nama & Cop Ketua Penyelidik

CATATAN : * Jika Laporan Akhir Penyelidikan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan ini perlu dikelaskan sebagai SULIT dan TERHAD.

**PARALLEL STRATEGIES ON A DISTRIBUTED
PARALLEL COMPUTER SYSTEM**

**(KAEDAH-KAEDAH SELARI PADA
SISTEM KOMPUTER SELARI TERAGIH)**

NORMA ALIAS

NORMA BINTI ALIAS

**RESEARCH VOTE NO:
75019**

*JABATAN MATEMATIK
FAKULTI SAINS*

**Jabatan Matematik
Fakulti Sains
Universiti Teknologi Malaysia**

2007

2007

PENGHARGAAN

Setinggi-tinggi penghargaan ditujukan kepada Pusat Pengurusan Penyelidikan, RMC, UTM dan Jabatan Matematik Fakulti Sains UTM di atas kerjasama sepanjang penyelidikan ini dijalankan. Ucapan ribuan terima kasih kepada para penyelidik-penyelidik yang telah memberikan sumbangan secara langsung dan tidak langsung dalam menyiapkan kajian tersebut.

Hanya Allah S.W.T yang akan membalas jasa murni tersebut yang telah disumbangkan oleh semua pihak. Semuga penyelidikan ini mendapat keberkatan dan akan diteruskan pada masa akan datang dalam mencapai objektif secara global.

Wasalam

ABSTRACT

The formulation of a new parallel iteration methods are created to solve the parabolic equations in one, two and three dimensions run on a distributed parallel computer systems on the homogeneous parallel machines with 20 PC Intel Pentium IV, speed 1.6MHz and with PVM application platform. The development of IADE class and AGE with (4,2) and (2,2) accuracies are oriented by ADI algorithm with time level splitting strategy by an alternating way. The alternative iterative method for IADE and AGE classes is created. There are IADEI method which is based on Richardson's formula and AGE_BRIAN which is based on linear interpolation concept. The development of a few strategies parallel is being as a mechanism for IADE to make it implemented in parallel. The Comparisons of the sequential performance measurements in IADE class with SUB, SOR, RB, MULTI, VECTOR and MF strategies are implicated the higher convergent and accuracy of IADEB-SUB and IADE-SUB methods. The Comparisons of the parallel performance measurements for IADE and AGE classes shown the elements of AGE class such as the overlapping subdomain and implicit block (2X2) are implicated the speedup and efficiency of AGE class is higher than IADE class. the minimum of the cost communication for the AGE class compared to IADEB-SUB and IADE-SUB are proved that AGE class is compatibility implemented on distributed parallel computer systems. The Expansion of AGE class on one, two and three dimensions for Cartesian, cylindrical and spherical coordinate systems make a decision that AGE-BRIAN methods is an alternative for AGE class in terms of convergent, accuracy, time executions, speedup, efficiency, effectiveness and temporal performance Communication activities and the consistent and work balance of data decomposition technique for CG method compatibility of the distributed parallel computer systems barriers.

ABSTRAK

Kajian penyelidikan ini mencakupi tuntutan kepada pembinaan teknik algoritma selari yang terkini dalam menyelesaikan model masalah bersaiz besar bagi satu sistem persamaan linear. Penganalisan pengiraan berangka untuk pelbagai kelas kaedah lelaran merupakan satu motivasi kepada pembinaan algoritma selari menerusi teknik partisi blok dapat mengurangkan penggunaan ruang lokasi ingatan dan kos komunikasi pada sistem komputer selari multipemproses ingatan teragih. Perumusan beberapa kaedah lelaran terbaru secara berjujukan dan selari dilaksanakan dalam menyelesaikan beberapa persamaan parabolik yang diimplementasikan pada sistem komputer selari yang dibangunkan oleh 20 pemproses Intel Pentium IV, berkelajuan 1.6GHz dengan aplikasi perisian penghantar utusan PVM. Kaedah lelaran tersebut didapati menumpu dengan kestabilan tidak bersyarat dan memiliki peringkat kejituan yang tinggi serta bercirikan domain penyelesaian berangka tak tersirat. Penciptaan kaedah lelaran variasi baru dalam kelas IADE dan AGE adalah berasaskan kepada strategi belahan TAS. Ujikaji berangka bagi model masalah multidimensi membuktikan kecepatan kadar penumpuan dan kejituan kaedah lelaran tersebut. Ke arah keberkesanan implementasi algoritma selari, pelbagai stratesi selari yang dicadangkan. Perbandingan ukuran prestasi algoritma selari dengan pelbagai strategi selari membuktikan kadar penumpuan dan masa pelaksanaan algoritma berjujukan bagi kelas IADE adalah dalam turutan SUB, SOR, RB, MULTI, VECTOR dan MF. Analisis ukuran prestasi algoritma selari mendapati kelas AGE mengatasi kelas IADE selari dari segi kecepatan kadar penumpuan berdasarkan kepada faktor kemampuan pemproses melaksanakan tugas saling tak bersandar, implikasi daripada subdomain tak bersandar, keoriginalan kaedah blok tersirat yang ditransformasikan secara terus kepada subdomain tak tersirat. Perbandingan ukuran prestasi algoritma selari ke atas kedua-dua kelas ini juga membuktikan kos komunikasi kaedah AGE adalah kurang daripada kelas IADE. Implikasinya, kesesuaian implementasi kelas AGE pada sistem komputer selari ingatan teragih juga dibincangkan. Penemuan ini merupakan satu titik tolak kepada peningkatan kadar penumpuan, kejituan, masa pelaksanaan, kecekapan, keberkesanan dan prestasi sementara jika kelas AGE dikembangkan kepada aplikasi model masalah multidimensi bagi sistem koordinat kartesian dan koordinat kutub dan seterusnya melahirkan pembinaan BRIAN dengan variasi baru sebagai alternatif kepada DOUGLAS. Aktiviti komunikasi dan teknik pengagihan data dipertimbangkan dalam mencakupi kekangan-kekangan pada sistem komputer selari ingatan teragih.

CONTENTS

	PAGE
Title	i
Acknowledgement	ii
Abstract	iii
Abstrak	iv
Contents	v
List of Tables	vii
List of Figures	viii
List of Acronyms	ix
Chapter 1 Introduction	1
1.1 High Performance Computing	1
1.2 Single Instruction, Multiple Data (SIMD)	2
1.2.1 General Characteristics	2
1.2.2 Advantages	3
1.2.3 Disadvantages	3
1.3 The development of a parallel computer program	5
1.3.1 Four steps	5
1.3.2 Processes of parallel programming	6
Chapter 2 Literature	7
2.1 IADE Methods	7
2.2 AGE Methods	7
2.3 Some Numerical methods	8
Chapter 3 Methodology	9

3.1 Sequential algorithms	9
3.2 Parallel IADE algorithms	11
3.2.1 IADEI-SUB	11
3.2.2 IADEI-RB	12
3.2.3 IADEI-SOR	12
3.2.4 IADEI-MULTI	13
3.2.5 IADEI-VECTOR	14
3.2.6 IADEI-MICHELL-FAIRWEATHER	14
3.3 Parallel AGE algorithms	15
3.3.1 DOUGLAS Algorithms	15
3.3.2. BRIAN method	15
3.3.3 Gauss-Seidel method	16
3.4 Model Problems	18
3.4.1 One dimensional problem	18
3.4.2 Two dimensional problem	19
3.5 Parallel implementation	19
3.5.1 IADE Methods	19
3.5.2. AGE Methods	20
Chapter 4 Computational Results	21
4.1 Parallel performance measurement	21
4.2 IADE Methods	22
4.3 AGE Methods	27
Chapter 5 Concluding Remarks and suggestions	34
5.1 IADE Methods	34
5.2 AGE Methods	34
REFERENCES	36

LIST OF TABLES

Table No.	Title	Page
1	Performance measurements of the parallel strategies of IADEI methods.	24
2	Computational complexity for iteration for the parallel strategies of the IADEI methods.	27
3	Performance measurements of the sequential BRIAN and DOUGLAS methods	30
4	Computational complexity for parallel BRIAN and DOUGLAS methods	31
5	Communication cost for parallel BRIAN and DOUGLAS methods	31
6	Execution, communication and idle times (μ sec) for parallel BRIAN and DOUGLAS methods	32
7	Computational / communication ratio for the parallel BRIAN and DOUGLAS methods	32
8	Performance measurements vs. number of processors for the parallel BRIAN and DOUGLAS methods	33

LIST OF FIGURES

Figure no.	Title	Page
1.1	Distributed parallel computer systems under configuration	4
3.1	Domain decomposition technique for arithmetic operations	9
3.2	Minimizes the communication cost between the grids	11
3.3	Data exchange for the boarder line among the tasks	13
3.4	Region for one dimensional PDE problems	18
4.1	Idle time in communication process	22
4.2	The execution time vs. number of processors	24
4.3	The speedup vs. number of processors	25
4.4	The efficiency vs. number of processors	25
4.5	The effectiveness vs. number of processors	26
4.6	Temporal performance vs. number of processors	26
4.7	The speedup vs. number of processors	29
4.8	The efficiency vs. number of processors	29
4.9	The effectiveness vs. number of processors	30

LIST OF ACRONYMS

Acronym

SIMD	Single Instruction, Multiple Data
NUMA	Non-uniform Memory Access
IADE	Iterative Alternating Decomposition Explicit
AGE	Alternating Group Explicit
IADEI	Iterative Alternating Decomposition Explicit Interpolation
PVM	Parallel Virtual Machine

Chapter 1

Introduction

This research surveys new trends that will introduced a new approach in regards of solving a tridiagonal structure of matrices complex problems. This new loom will have a great major impact on all aspect of Mathematics and Science growth. It revealed under parallel computing technology in its evolution from chronological to parallel algorithmic practice. The exposure of iterative or domain decomposition approximate methods such as Jacobi and Gauss-Seidel Red Black, provides a consisted of guessing a value that initial guesses is to assume that they are all zero. Even some cases stated that Numerical method is useful; with some advantageous from Gauss Seidel Red Black's exposure, it had been the best method chosen in solving the parallelization algorithm of PDE problems. The experiment of parallel computing were carried out in more that 2 PC Intel Pentium IV under homogenous architecture platform of LINUX to perform the algorithm. The comparisons between sequential and parallel algorithm will be presented. The result of some computational experiments and the parallel performance in solving the equations will be discussed.

1.1 High Performance Computing

There are different ways to classify parallel computers. One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.

Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of *Instruction* and *Data*. Each of these dimensions can have only one of two possible states: *Single* or *Multiple*.

The matrix below defines the 4 possible classifications according to Flynn.

S I S D Single Instruction, Single Data	S I M D Single Instruction, Multiple Data
M I S D Multiple Instruction, Single Data	M I M D Multiple Instruction, Multiple Data

1.2 Single Instruction, Multiple Data (SIMD):

A type of parallel computer Single instruction: All processing units execute the same instruction at any given clock cycle multiple data: Each processing unit can operate on a different data element. This type of machine typically has an instruction dispatcher, a very high-bandwidth internal network, and a very large array of very small-capacity instruction units, Best suited for specialized problems characterized by a high degree of regularity, such as image processing. Synchronous (lockstep) and deterministic execution two varieties: Processor Arrays and Vector Pipelines

Examples (some extinct):

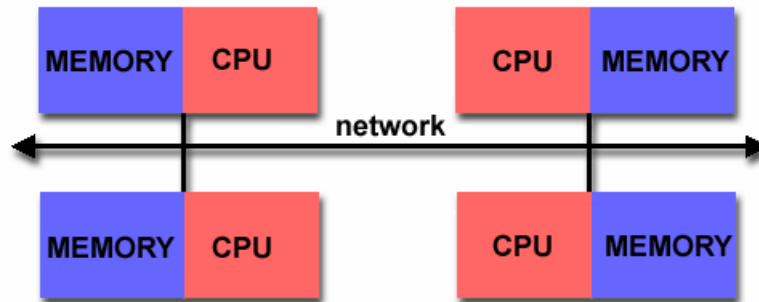
Processor Arrays: Connection Machine CM-2, Maspar MP-1, MP-2

Vector Pipelines: IBM 9000, Cray C90, Fujitsu VP, NEC SX-2, Hitachi S820

Distributed Memory

1.2.1 General Characteristics

Like shared memory systems, distributed memory systems vary widely but share a common characteristic. Distributed memory systems require a communication network to connect inter-processor memory.



Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors. Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.

When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.

The network "fabric" used for data transfer varies widely, though it can be as simple as Ethernet.

1.2.2 Advantages

Memory is scalable with number of processors. Increase the number of processors and the size of memory increases proportionately. Each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherency. Cost effectiveness: can use commodity, off-the-shelf processors and networking.

1.2.3 Disadvantages

The programmer is responsible for many of the details associated with data communication between processors. It may be difficult to map existing data structures, based on global memory, to this memory organization. Non-uniform memory access (NUMA) times

According to Wikipedia, the free encyclopedia in Internet, the definitions of Parallel Computing is the simultaneous execution of the *same task* (split up and specially adapted) on multiple processors in order to obtain results faster. The idea is based on the fact that the process of solving a problem usually can be divided into smaller tasks, which may be carried out simultaneously with some coordination.

With the idea of 20 PCs, 20 GB Hard Disk (each), 1.67 MHz processors (each) and using Intel Pentium IV; to obtaining the fastest result in solving PDE can be resolved with this machine.



Figure 1.1: Distributed parallel computer systems under consideration

This research encompasses studies on recent advances in development of the parallel algorithmic techniques to solve a large-scale problems involving the solution of The Partial Differential Equations problems had been discussed with a few methods that are; the implementation of domain decomposition technique with introduces of $[A] = [L] [U]$.

1.3 The development of a parallel computer program

There are a few steps to develop the parallel computer programs.

1.3.1 Four steps

There are 4 steps in developing a parallel program, which is done by student with understanding some important concepts such as task, process and processors

1. Decomposition
2. Assignment
3. Algometric
4. Mapping

Firstly, computational complexity is break up into task to be divided among process and the gold of this part is to make sure there are enough tasks to keep processes busy. The realization of the finite element method on parallel computers is usually based on a domain decomposition approach. Student is concerned with the problem of finding an optimal decomposition and an appropriate mapping of the sub-domains to the processors. The quality of this partitioning is measured in several metrics but it is also expressed in the computing time for solving specific systems of grand challenge applications. . In particular, the data structure and the accumulation algorithm are introduced. Then students are suggested to develop several partitioning algorithms to make the comparison in term of performance.

Secondly, specifying mechanism to divide work among processes statically or dynamically assigned. The function of this part is to maximize work balancing, reduce communication activities and management cost.

Thirdly is structuring communication, organizing data and scheduling tasks. The goals are to reduce cost of communication and synchronization, schedule tasks to satisfy dependency and reduce overhead of parallelism management. Lastly, mapping the process into particular processors based on network topology.

1.3.2 Processes of parallel programming

The processes in creating a parallel program are as follows:

1. Assumption that the sequential algorithm is given.
2. Identify work that can be done in parallel
3. Partition work and data among processes
4. Manage data access, communication activities and the synchronization

Chapter 2

Literature

2.1 IADE Methods

Six strategies of parallel algorithms are implemented to exploit the convergence of IADE. In the domain decomposition strategy the IADE Michell-Fairweather which is fully explicit is derived to produce the approximation of grid- i and not totally dependent on the grid $(i-1)$ and $(i+1)$. In IADE Red Black and IADE SOR strategies, the domain is decomposed into two different subdomains. The concept of multidomain is observed in the IADE Multicoloring method. The decomposition of domain split into w different groups of domain. On the vector iteration strategy, parallel IADE is run in two sections. This method converges if the inner convergence criterion is achieved for each section.

2.2 AGE Methods

In Evans & Sahimi (1989) the discretization of parabolic partial differential equation is derived from Iterative Alternating Decomposition Explicit Method (IADE) and Alternating Group Explicit Method (AGE). In Sahimi & Alias (2000) the six strategies of parallel algorithms in solving parabolic partial differential equations is implemented. These strategies were found to be more effective using a distributed memory machines. In Alias, Sahimi & Abdullah (2002), the Conjugate Gradient on AGE method (AGE-CG) was found to be more convergence and accurate compared to AGE. In this paper, the computational analysis of the proposed strategies is presented by using explicit block (2×2) . These schemes can be effective in reducing data storage accesses in distributed memory and communication time on a distributed computer systems. The research focuses on the cost communication and computational complexity for these iterative methods. All the parallel strategies have been developed to run on a cluster of workstations based on Parallel Virtual Machine environment (Geist .el, 1994).

2.3 Some Numerical methods

In this research, Alternating Group Explicit Method (AGE) for solving two dimensional heat model is compared with Alternating Group Explicit - Interpolation Method. This new iterative algorithm has been developed based on (2x2) block schemes expressed in explicit point form notation. In Evans & Sahimi (1989) the discretization of parabolic partial differential equation is derived from (AGE). In Alias, Sahimi & Abdullah (2002), the Conjugate Gradient on AGE method (AGE-CG) was found to be more convergence and accurate than AGE.

Chapter 3

Methodology

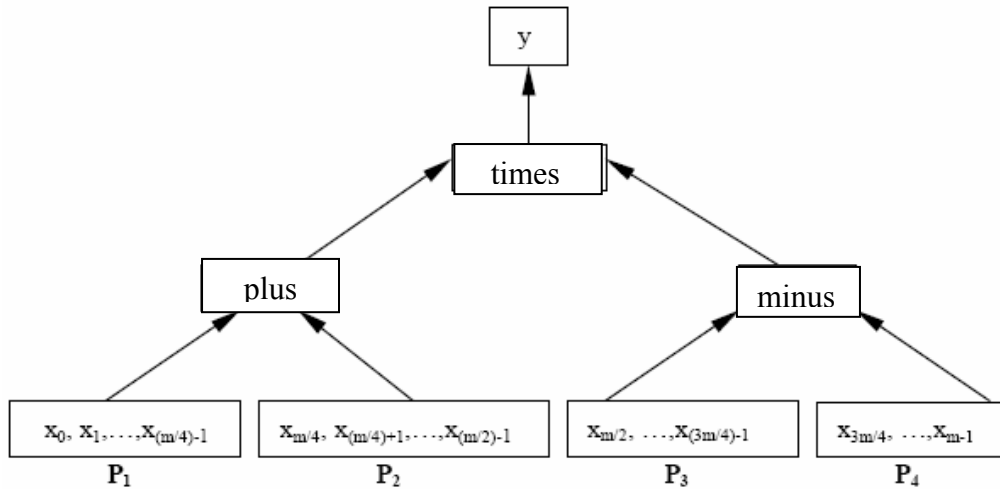


Figure 3.1: Domain decomposition technique for arithmetic operations.

Parallel strategies under consideration are focuses on domain decomposition technique as shown in figure above. The figure shows the examples of domain decomposition technique for arithmetic operations.

3.1 Sequential algorithms

The new technique known as Iterative Alternating Decomposition Explicit Interpolation

$$\begin{aligned}
& \text{at time level } (k + \frac{1}{2}) \\
& (I + \alpha \mathbf{G}_1)u^{(k+\frac{1}{2})} = (I + (\alpha + 2r)\mathbf{G}_1)(I + 2r\mathbf{G}_2) + \beta \mathbf{G}_1 \mathbf{G}_2 u^{(k)} - 2rf \\
& \text{at time level } (k + 1) \\
& (I + \alpha \mathbf{G}_2)u^{(k+1)} = u^{(k+\frac{1}{2})} + \alpha \mathbf{G}_2 u^{(k)} \\
& \text{with } \alpha = \frac{1}{12} - \frac{2}{3}r \text{ and } \beta = \frac{2r(3\nu - 2r)}{3}
\end{aligned}$$

The coefficient matrix \mathbf{A} in equation (3) however is decomposed into,

$$\mathbf{A} = \mathbf{G}_1 + \mathbf{G}_2 + \frac{1}{6}\mathbf{G}_1 \mathbf{G}_2$$

Method (IADEI) is applied to linear system in the Parabolic equation. The sequential and the parallel strategies of IADEI algorithms are described in detail and the numerical results are presented. Iterative Alternating Decomposition Explicit Interpolation Method (IADEI) is based on Iterative Alternating Decomposition Explicit (Sahimi, 1989) and the modification of matrix \mathbf{A} . The discretization of IADEI method is obtained in the implicit and explicit forms as follows,

i. at time level $(k + \frac{1}{2})$

$$\begin{aligned}
u_i^{(k+\frac{1}{2})} &= \frac{1}{A}(s_{i-1}u_{i-1}^{(k)} + v_i u_i^{(k)} + s s u_{i+1}^{(k)} - w_{i-1}u_{i-1}^{(k+\frac{1}{2})} - DDf_i), \quad i = 1, 2, \dots, m, \\
s_0 &= v_0 = w_0 = 0 \text{ and } A \neq 0, \quad \forall i \in [1, m].
\end{aligned} \tag{7}$$

ii. At time level $(k + 1)$

$$\begin{aligned}
u_{m+1-i}^{(k+1)} &= \frac{1}{1 + d_{m+1-i}}(u_{m+1-i}^{(k+\frac{1}{2})} + d_{m+1-i}u_{m+1-i}^{(k)} + d d u_{m+2-i}^{(k)} - d d u_{m+2-i}^{(k+1)}) \\
&\text{with } d_i \neq 0 \text{ and } \forall i \in [1, m]
\end{aligned} \tag{8}$$

3.2 Parallel IADE algorithms

The objectives of the parallel algorithms are to minimize the communication cost and computational complexity.

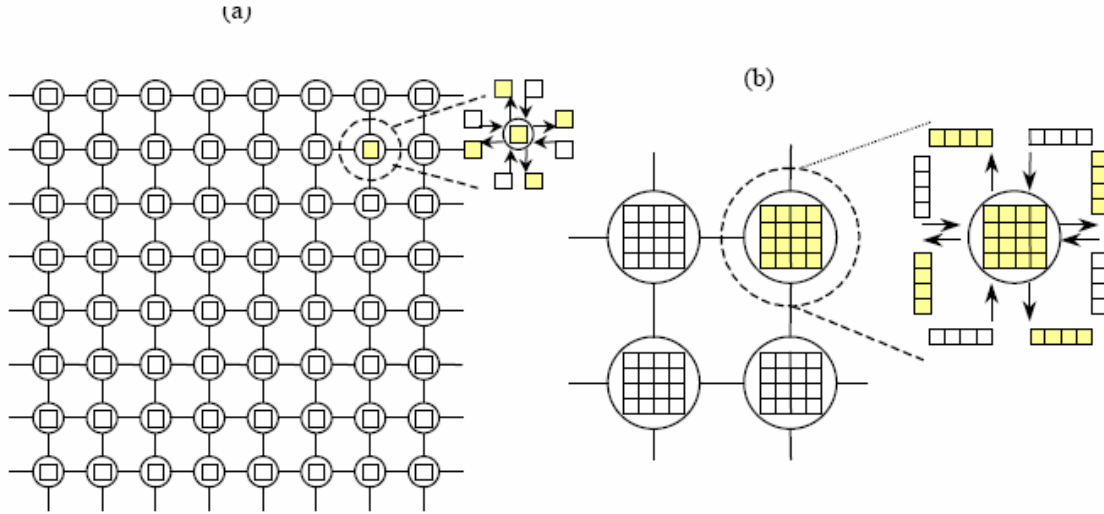


Figure 3.2: Minimizes the communication cost between the grids.

The sequential algorithm for IADEI shown that the approximation solution for grid u_i is depend on u_{i-1} and the approximation solution for u_{m+1-i} is depend on u_{m+2-i} . To avoid dependently situation, some parallel strategies is developed to create the non-overlapping subdomains.

3.2.1 IADEI-SUB

On the strategy of Incomplete Block LU preconditioners on slightly non-overlapping subdomains, the domain is decomposed into p processors with incomplete subdomain. This strategy implemented the incomplete factorization with parameter of algebraic boundary condition as follows,

i. at time level $(k + \frac{1}{2})$

$$\begin{aligned} & Au_{i-1}^{(k+\frac{1}{2})} - s_{i-1}u_{i-1}^{(k)} - v_{i-1}u_{i-1}^{(k)} - ssu_i^{(k)} + \beta w_{i-2}u_{i-2}^{(k+\frac{1}{2})} - \beta w_{i-1}u_{i-1}^{(k+\frac{1}{2})} \\ & = -DDf_{i-1}), i \in \overline{\Omega}. \end{aligned}$$

i. at time level $(k + 1)$

$$\begin{aligned} & (1 + d_{i+1})u_{i+1}^{(k+1)} - (u_i^{(k+\frac{1}{2})} + d_{i+1}u_{i+1}^{(k+1)} + ddu_{i+2}^{(k)} - ddu_{i+2}^{(k+1)}) = 0, i \in \overline{\Omega}. \\ & i \in \overline{\Omega}. \end{aligned}$$

3.2.2. IA DEI-RB

In IA DEI-RB strategy, the domain is decomposed into two different subdomains. There are the approximate solution on the odd grids and even grids. Computation on odd grids is executed followed by even grids. These two subdomains are not dependent on each other. IA DEI-RB is run in parallel for each subdomain in alternating way on two time steps. The parallel IA DEI-RB formula is as follows,

i) at time level $(k + \frac{1}{2})$

$$\begin{aligned} u_i^{(k+\frac{1}{2})} &= u_i^{(k+\frac{1}{2})} + \frac{\omega}{A}(-Au_i^{(k+\frac{1}{2})} + s_{i-1}u_{i-1}^{(k)} + v_iu_i^{(k)} + ssu_{i+1}^{(k)} - w_{i-1}u_{i-1}^{(k+\frac{1}{2})} \\ &\quad -DDf_i), i \in \Omega^R \\ u_i^{(k+\frac{1}{2})} &= u_i^{(k+\frac{1}{2})} + \frac{\omega}{A}(-Au_i^{(k+\frac{1}{2})} + s_{i-1}u_{i-1}^{(k)} + v_iu_i^{(k)} + ssu_{i+1}^{(k)} - w_{i-1}u_{i-1}^{(k+\frac{1}{2})} \\ &\quad -DDf_i), i \in \Omega^H \end{aligned}$$

ii) at time level $(k + 1)$

$$\begin{aligned} u_{m+1-i}^{(k+1)} &= \frac{\omega}{1 + d_{m+1-i}}(-(1 + d_{m+1-i})u_{m+1-i}^{(k+1)} + u_{m+1-i}^{(k+\frac{1}{2})} + d_{m+1-i}u_{m+1-i}^{(k)} + ddu_{m+2-i}^{(k)} \\ &\quad -ddu_{m+2-i}^{(k+1)}), i \in \Omega^R \\ u_{m+1-i}^{(k+1)} &= \frac{\omega}{1 + d_{m+1-i}}(-(1 + d_{m+1-i})u_{m+1-i}^{(k+1)} + u_{m+1-i}^{(k+\frac{1}{2})} + d_{m+1-i}u_{m+1-i}^{(k)} + ddu_{m+2-i}^{(k)} \\ &\quad -ddu_{m+2-i}^{(k+1)}), i \in \Omega^H \end{aligned}$$

3.2.3. IA DEI-SOR

Using the well-known fact of the IA DEI-RB, the parallel algorithm for IA DEI-CG-SOR takes the form similar to IA DEI-RB but the acceleration parameter was chosen to provide the most rapid convergence

3.2.4 IADEI-MULTI

Multicolor technique has been used extensively for the solution of the large-scale problems of linear system of equations on parallel and vector computer (Ortega, 1987). By the definition of multidomain, the domain is decomposed into w different groups. IADEI-MULTI is an advanced concept of IADEI-RB. Typically, one chooses the minimum number of colors w so that the coefficient matrix takes the block form. In particular $w=2$, then IADEI-MULTI is the IADEI-RB. The Domains for colors 1, 2, 3, ..., w are noted as w_1, w_2, \dots, w_w . The subdomains w_i are distributed into different groups of grid. In the process of assignment, subdomains w_i are mapped into the processors p in the alternating way. At each time step, the computational grid for domain started its execution with w_1 , followed by w_2 and ends with w_w . The IADEI-MULTI allows the possibility of a high degree of parallelism and vectorization. However, IADEI-MULTI, as opposed to the natural ordering, may have a deleterious effect on the rate of convergence.

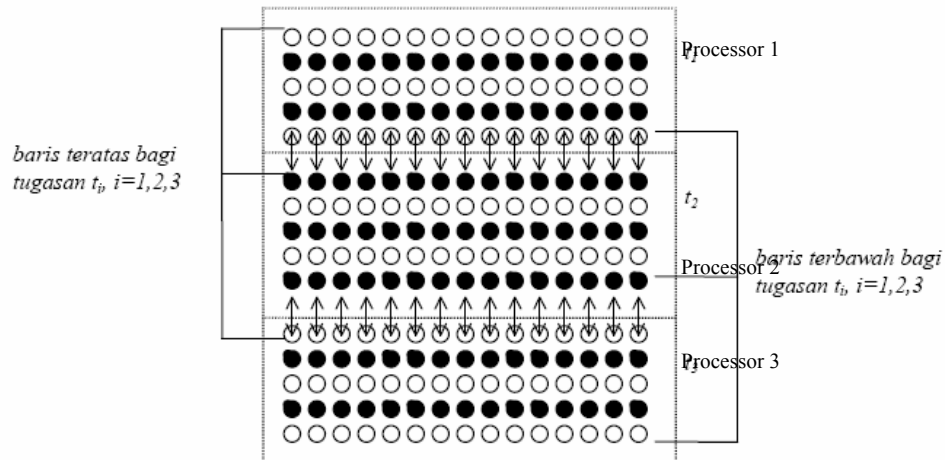


Figure 3.3 : Data exchange for the boarder line among the tasks

3.2.5 IA DEI-VECTOR

On the vector iteration strategy, the parallel IA DEI-VECTOR is run in two convergence sections. The first section is at time level $(k+1/2)$ and the second section is time level $(k+1)$. This method converges if the inner convergence criterion is achieved for each section. The inner convergence criterions are definite global convergence criterion.

3.2.6 IA DEI-MICHELL-FAIRWEATHER

The IA DEI-Michell-Fairweather (IA DEI-MF), which is fully explicit, is derived to produce the approximation of grid- i and which is not totally dependent on the grid $(i-1)$ and $(i+1)$. The approximation at the first and second intermediate levels are computed directly. The explicit form of equation are given by,

$$u_i^{(k+\frac{1}{2})} = \sum_{l=1}^{i-1} \frac{(-1)^T v^{i-l-1} \prod_{j=l}^{i-2} k_j (E k_{i-1} + G e_{i-1} k_{i-1})}{\prod_l^{i-1} A} u_l^{(k)} \\ + \sum_{l=1}^i \frac{(-1)^H v^{i-l} \prod_{j=l}^{i-1} k_j (J + H e_i + G(k_{i-1} h + e_i))}{\prod_l^i A} u_l^{(k)} \\ + \sum_{l=1}^{i+1} \frac{(-1)^T v^{i-k+l} \prod_{j=l}^i k_j (H h + G h)}{\prod_l^{i+1} A} u_l^{(k)} + \sum_{l=1}^i \frac{(-1)^H v^{i-k} \prod_{j=l}^{i-1} k_j f_k}{\prod_l^i A} u_l^{(k)}$$

$$H = \begin{cases} l+1, & i = 1, 3, 5, \dots, m-1 \\ l, & i = 2, 4, 6, \dots, m \end{cases} \quad T = \begin{cases} l, & i = 1, 3, 5, \dots, m-1 \\ l+1, & i = 2, 4, 6, \dots, m \end{cases}$$

with $L = 2r$, $E = \alpha + L$, $G = L(\alpha + L) + \beta$ dan $J = 1 + \alpha + L$.
at time level $(k+1)$,

$$u_m^{(k+1)} = \frac{\alpha e_m}{t_m} u_m^{(k)} + \frac{u_m^{(k+\frac{1}{2})}}{t_m} \\ u_i^{(k+1)} = \frac{\alpha e_i}{d_i} u_i^{(k)} + \alpha \sum_{l=i}^{m-1} \frac{(-1)^H \prod_{j=i}^l h}{\prod_{j=i}^l t_j} u_{l+1}^{(k)} + \alpha \sum_{l=i}^{m-1} \frac{(-1)^T \prod_{j=i}^l h e_{l+1}}{\prod_{j=i}^{l+1} t_j} u_{l+1}^{(k)} \\ + \sum_{l=i}^{m-1} \frac{(-1)^T h_j^{l-i+1}}{\prod_{j=i}^{l+1} t_j} u_{l+1}^{(k+\frac{1}{2})}$$

$$H = \begin{cases} l+1, & i = 1, 3, 5, \dots, m-1 \\ l, & i = 2, 4, 6, \dots, m \end{cases} \quad T = \begin{cases} l, & i = 1, 3, 5, \dots, m-1 \\ l+1, & i = 2, 4, 6, \dots, m \end{cases}$$

3.3 Parallel AGE algorithms

The parallel strategies is based on straightforward concept to minimizes the communication and computational costs.

3.3.1 DOUGLAS Algorithms

DOUGLAS Algorithms is based on the Douglas - Rachford formula for AGE fractional scheme (Sahimi, 1989) takes the form,

$$(\mathbf{G}_1 + r\mathbf{I})u_{(r)}^{(k+\frac{1}{4})} = (r\mathbf{I} - \mathbf{G}_1 - 2\mathbf{G}_2 - 2\mathbf{G}_3 - 2\mathbf{G}_4)u_{(r)}^{(k)} + 2\mathbf{f} \quad (1)$$

$$(\mathbf{G}_2 + r\mathbf{I})u_{(r)}^{(k+\frac{1}{2})} = \mathbf{G}_2u_{(r)}^{(k)} + ru_{(r)}^{(k+\frac{1}{4})} \quad (2)$$

$$(\mathbf{G}_3 + r\mathbf{I})u_{(r)}^{(k+\frac{3}{4})} = \mathbf{G}_3u_{(r)}^{(k)} + ru_{(r)}^{(k+\frac{1}{2})} \quad (3)$$

$$(\mathbf{G}_4 + r\mathbf{I})u_{(r)}^{(k+1)} = \mathbf{G}_4u_{(r)}^{(k)} + ru_{(r)}^{(k+\frac{3}{4})} \quad (4)$$

Where \mathbf{A} is the sum of its constituent symmetric and positive define matrices \mathbf{G}_1 , \mathbf{G}_2 , \mathbf{G}_3 and \mathbf{G}_4 ,

$$\mathbf{A} = \mathbf{G}_1 + \mathbf{G}_2 + \mathbf{G}_3 + \mathbf{G}_4 \quad (5)$$

3.3.2. BRIAN method

BRIAN method is based on the AGE algorithm with Douglas - Rachford variant and linear interpolation (BRIAN) concepts using the fractional strategy. BRIAN algorithm has been developed as an alternative to the parallel and sequential algorithm of DOUGLAS method. The formula for BRIAN method for 2-dimensional problem leads to five intermediate levels are as follows,

$$(\mathbf{G}_1 + r\mathbf{I})u_{(r)}^{(k+\frac{1}{5})} = (r\mathbf{I} - \mathbf{G}_2 - \mathbf{G}_3 - \mathbf{G}_4)u_{(r)}^{(k)} + \mathbf{f} \quad (6)$$

$$(\mathbf{G}_2 + r\mathbf{I})u_{(r)}^{(k+\frac{2}{5})} = \mathbf{G}_2u_{(r)}^{(k)} + ru_{(r)}^{(k+\frac{1}{5})} \quad (7)$$

$$(\mathbf{G}_3 + r\mathbf{I})u_{(r)}^{(k+\frac{3}{5})} = \mathbf{G}_3u_{(r)}^{(k)} + ru_{(r)}^{(k+\frac{2}{5})} \quad (8)$$

$$(\mathbf{G}_4 + r\mathbf{I})u_{(r)}^{(k+\frac{4}{5})} = \mathbf{G}_4u_{(r)}^{(k)} + ru_{(r)}^{(k+\frac{3}{5})} \quad (9)$$

and with linear interpolation, we obtain,

$$u_{(r)}^{(k+1)} = 2u_{(r)}^{(k+\frac{4}{5})} - u_{(r)}^{(k)} \quad (10)$$

3.3.3 Gauss-Seidel method

Gauss-Seidel method is chosen as a control schemes. The Gauss-Seidel method is a technique used to solve a linear system of equations. The method is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel. The method is similar to the Jacobi method (and likewise diagonal dominance of the system is sufficient to ensure convergence, meaning the method will work).

We seek the solution to a set of linear equations, expressed in matrix terms as

$$A\phi = b.$$

The Gauss-Seidel iteration is

$$\phi^{(k+1)} = (D - L)^{-1} (U\phi^{(k)} + b),$$

where D , $-L$, and $-U$ represent the diagonal, lower triangular, and upper triangular parts of the coefficient matrix A and k is the iteration count. This matrix expression is mainly of academic interest, and is not used to program the method. Rather, an element-based approach is used:

$$\phi_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij}\phi_j^{(k+1)} - \sum_{j>i} a_{ij}\phi_j^{(k)} \right), \quad i = 1, 2, \dots, n.$$

Note that the computation of $\phi_i^{(k+1)}$ uses only those elements of $\phi^{(k+1)}$ that have already been computed and only those elements of $\phi^{(k)}$ that have yet to be advanced to iteration $k + 1$. This means that no additional storage is required, and the computation can be done in place ($\phi^{(k+1)}$ replaces $\phi^{(k)}$). While this might seem like a rather minor concern, for large systems it is unlikely that every iteration can be stored. Thus, unlike the Jacobi method, we do not have to do any vector copying should we wish to use only one storage vector. The iteration is generally continued until the changes made by an iteration are below some tolerance.

Algorithm

Chose an initial guess ϕ^0

for $k := 1$ step 1 until convergence do

 for $i := 1$ step 1 until n do

$\sigma = 0$

 for $j := 1$ step 1 until $i-1$ do

$$\sigma = \sigma + a_{ij}\phi_j^{(k)}$$

 end (j-loop)

 for $j := i+1$ step 1 until n do

$$\sigma = \sigma + a_{ij}\phi_j^{(k-1)}$$

 end (j-loop)

$$\phi_i^{(k)} = \frac{(b_i - \sigma)}{a_{ii}}$$

 end (i-loop)

check if convergence is reached

end (k-loop)

3.4 Model Problems

3.4.1 One dimensional problem

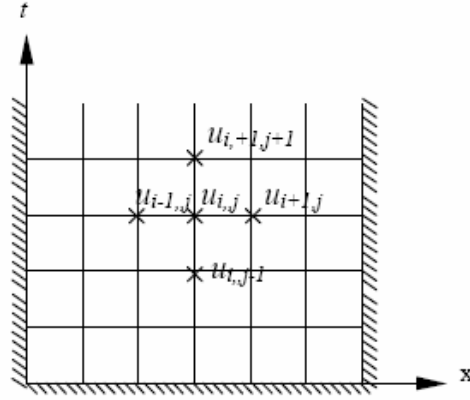


Figure 3.4 : Region for one dimensional PDE problems

Equation 0.0.1 is discretized by the finite difference formula and leads to the generalized approximation stencil as,

$$\begin{aligned}
 -\lambda\theta u_{i-1,j+1} + (1 + 2\lambda\theta)u_{i,j+1} - \lambda\theta u_{i+1,j+1} &= \lambda(1 - \theta)u_{i-1,j} + [1 - 2\lambda(1 - \theta)]u_{i,j} \\
 &+ \lambda(1 - \theta)u_{i+1,j}
 \end{aligned} \tag{0.0.2}$$

where $i = 1, 2, 3, \dots, m$, $j = 1, 2, 3, \dots, T$ and $\lambda = \frac{\Delta t}{(\Delta x)^2}$ leads to the a large and sparse linear system of equation,

$$\mathbf{A}u = \mathbf{f} \tag{0.0.3}$$

The parallel iteration algorithms for solving a large and sparse linear system 0.0.3 has been developed and implemented successfully by Alternating Group Explicit Method (AGE), Alternating Group Explicit-Conjugate Gradient Method (AGE_CG), Gauss Seidel Red Black Method (GSRB) and Gauss Seidel Red Black-Conjugate Gradient Method (GSRB_CG)

The model problem under consideration is one dimensional parabolic equation (Smith, 1979).

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2}, \quad 0 \leq x \leq 1, \quad 0 < t, \quad (0.0.1)$$

with initial condition, $U(x, 0) = f(x)$, $0 \leq x \leq 1$.

Boundary condition, $U(0, t) = g(t)$, $0 < t \leq T$ and $U(1, t) = h(t)$, $0 < t \leq T$.

Where $f(x) = \sin(\pi x)$, $0 \leq x \leq 1$, and $g(t) = h(t) = 0$, $0 < t \leq 1$.

and satisfying the exact solution of equation, $U(x, t) = e^{-\pi^2 t} \sin(\pi x)$

Let's consider equation (0.0.1) on Ω be the domain of $0 \leq x \leq 1$ and $0 < t \leq 1$ with grid spacing $\Delta x = \frac{1}{m} = h$ in both directions $x_i = x_0 + ih$ and $t_j = t_0 + jh$ for $\forall i, j = 0, 1, 2, \dots, m$.

3.4.2 Two dimensional problem

The parallel strategies were tested also on the 2-dimesional parabolic partial differentials equation as follows (Gourlay & MacGuire, 1971),

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + h(x, y, t),$$

$$(x, y, t) \in R \times (0, T] \quad , \quad (11)$$

subject to the initial condition

$$U(x, y, 0) = f(x, y), \quad (x, y, t) \in R \times 0$$

and the boundary conditions

$$U(x, y, t) = G(x, y, t), \quad (x, y, t) \in \partial R \times (0, T]$$

where $h(x, y, t) = \sin x \sin y e^{-t} - 4$,

$0 \leq x \leq 1$, $0 \leq y \leq 1$, $0 \leq t$

and $U(x, y, 0) = \sin x \sin y + x^2 + y^2$.

The theoretical solution is given by,

$$U(x, y, t) = (\sin x \sin y) e^{-t} + x^2 + y^2$$

$$0 \leq x \leq 1, \quad 0 \leq y \leq 1, \quad 0 \leq t$$

3.5 Parallel implementation

3.5.1 IADE Methods

All the parallel strategies are based on the non-overlapping subdomain. There is no data exchange between the neighboring processors at the iteration (k) but there are inter-processor communications between the iteration (k) and the next iteration (k+1). A

typical parallel implementation of a parallel IADEI assigns several mesh points to each processor p such that each processor only communicates with its two nearest neighbors. The computations of the approximation solutions in subdomain w_p are executed independently. The stopping criteria in the processors p are investigated by measuring the size of the inner residuals. Let us define the residual computed in the processors p . This quantity is kept in the processor's memory between successive iterations and it is checked if the residual is reduced by the convergence criterion. The master processor checked the maximum of local convergence criterion and the iteration stopped when the global convergence criterion is met.

3.5.2. AGE Methods

As the AGE class is Fully explicit its feature can be fully utilized for parallesation. Firstly, Domain is distributed to p subdomains by the master processor. The partitioning is based on data decomposition technique. Secondly, the subdomains p of BRIAN and DOUGLAS methods are assigned into processors in block ordering. The domain decomposition for BRIAN and DOUGLAS methods are implemented in four and five time level, respectively. The communication schemes between the slave processors are needed for the computations in the next iterations. The parallelization of BRIAN and DOUGLAS is achieved by assigning the explicit block (2x2) in this way and proved that the computations involved are independent between processors. The parallelism strategy is straightforward and the domain is distributed to non over lapping subdomain. Based on the limited parallelism, this scheme can be effective in reducing computational complexity and data storage accesses in distributed parallel computer systems. The iterative procedure is continued until convergence is reached.

Chapter 4

Computational Results

4.1 Parallel performance measurement

The following definitions are used to measure the parallel strategies, speedup $S_p = \frac{T_1}{T_p}$,

efficiency $C_p = \frac{S_p}{P}$, effectiveness $F_p = \frac{C_p}{T_p}$, and temporal performance $L_p = T_p^{-1}$, where

T_1 is the execution time on one processor, T_p is the execution time on p processors and the unit of L_p is work done per microsecond. The important factors effecting the performance in message passing on distributed memory computer systems are communication patterns and computational per communication ratios.

The important factors affecting the performance in message-passing paradigm on a distributed memory computer systems are communication patterns and computational/communication ratios. The communication time will depend on many factors including network structure and network contention (Wolfgang, 1988). Parallel execution time, t_{para} is composed of two parts, computation time (t_{comp}) and communication time (t_{comm}). t_{comp} is the time to compute the arithmetic operations such as multiplication and addition operations of sequential algorithms. Analysis of the t_{comp} assumes that all the processors are the same and the operating at the same speed. t_{comm} will depend upon the size of message. If the number of iterations b , and size of the message for communication m , the formula for communication time is as follows,

$$t_{comm} = b(t_{start} + m t_{data} + t_{idle})$$

where t_{start} is the startup time (message latency). t_{data} is time to send a message with no data. The term t_{data} is the transmission time to send one data word. t_{idle} is the time for message latency; time to wait for all the processors to complete the process as shown in the figure below. It is also a means of quantifying the degree of load imbalance in the parallel algorithm. In order to estimate the coefficients t_{start} and t_{data} , a number of experiments were conducted for different message sizes.

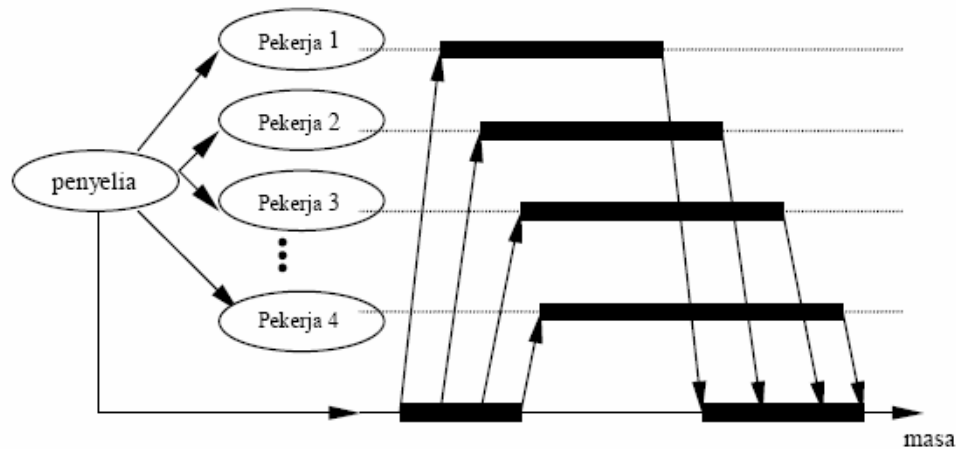


Figure 4.1 : Idle time in communication process

4.2 IADE Methods

The results obtained for the various parallel strategies of IADEI in table below. The worst performances are shown by IADEI-MF and IADEI-VECTOR. The sequential IADEI is better in accuracy and convergence than all the parallel strategies of IADEI. In comparison with the parallel strategies of IADEI, these results also show that the time execution for IADEI-SUB was about 2 times shorter than other parallel strategies. Furthermore, IADEI-SUB is the best in terms of convergence and accuracy.

The results present the numerical properties of the parallel solver on the homogeneous architecture of 20 PC systems with Linux operating, Intel Pentium IV processors, speedup 20GB HDD and connected with internal network Intel 10/100 NIC. The following definitions are used to measure the parallel strategies are speedup, efficiency, effectiveness and temporal performance. Parallel Gauss Seidel Red Black is chosen as the control scheme. The graph of the execution time, speedup, efficiency, effectiveness and the temporal performance versus number of the processors were plotted in Figures below. The algorithm of parallel strategies with the highest performance executed in the least time and is therefore the best algorithm. As expected, the execution time decreases with

the increasing p . IADEI-SUB strategy is found to give the best performance because of the minimum memory access and data sharing.

At $p=20$ processors, all the parallel strategies of IADEI yield approximately equal performance in speedup. The efficiency of IADEI-MF and IADEI-VECTOR strategies are decreased drastically. This is the result of the additional overhead imposed by having communications routed through the PVM daemon with high number of iterations.

The results have shown that the effectiveness of IADEI-SUB is superior to the IADEI-SOR, IADEI-RB and IADEI-MULTI for all numbers of processors. Usually, the temporal performance is used to compare the performance of different parallel algorithms. The temporal performance of the parallel strategies as in the following order,

1. IADEI-LU
2. IADEI-SOR
3. IADEI-RB
4. IADEI-MULTI
5. IADEI-VECTOR
6. IADEI-MF

Table 1 Performance measurements of the parallel strategies of IADEI methods

$m = 720003$, $\Delta x = 1.3889E^{-6}$, $\Delta t = 9.6450E^{-13}$, level=50 $t = 4.8225E^{-11}$ $\lambda = 0.5$, $\theta = 1.00$, $\epsilon = 1.0E^{-15}$

IADEI	SQENT	SUB	SOR	RB	MULTI	VEKT0R	MF	GBRB
t_{para}	43.0972	45.5205	103.873	114.8103	198.0183	291.0482	2106.659	156.4432
iterat.	261	301	358	360	360	261	261	600
rmse	$1.5921E^{-9}$	$1.5921E^{-9}$	$1.5567E^{-9}$	$1.5921E^{-9}$	$1.5921E^{-9}$	$1.5921E^{-9}$	$1.5921E^{-9}$	$1.5921E^{-9}$
$ r $	$6.6613E^{-16}$	$9.9920E^{-16}$	$6.1062E^{-16}$	$6.1062E^{-16}$	$8.8818E^{-12}$	$9.9920E^{-16}$	$6.6613E^{-16}$	$1.1123E^{-16}$
ave_rmse	$1.9846E^{-7}$	$1.9846E^{-7}$	$1.9846E^{-7}$	$1.9846E^{-7}$	$1.9846E^{-7}$	$1.9846E^{-7}$	$1.9846E^{-7}$	$1.9846E^{-7}$
r.maks	$5.3374E^{-17}$	$5.3374E^{-17}$	$5.3374E^{-17}$	$5.3314E^{-17}$	$5.3374E^{-17}$	$5.3374E^{-17}$	$5.3374E^{-17}$	$5.3374E^{-17}$
r	0.74	0.7	0.54	0.55	0.56	0.73	0.74	—
ω_y	—	—	1.01	1.0	1.02	—	—	—
ω_z	—	—	1.0	1.0	0.94	—	—	—
L_y	—	—	—	—	—	1036	—	—
L_z	—	—	—	—	—	987	—	—
ϵ_y	—	—	—	—	—	$1.0E^{-14}$	—	—
ϵ_z	—	—	—	—	—	$1.0E^{-15}$	—	—

rmse= root means square error, $|r|$ = absolute error, r.maks = maximum error and ave_rmse = average of rmse

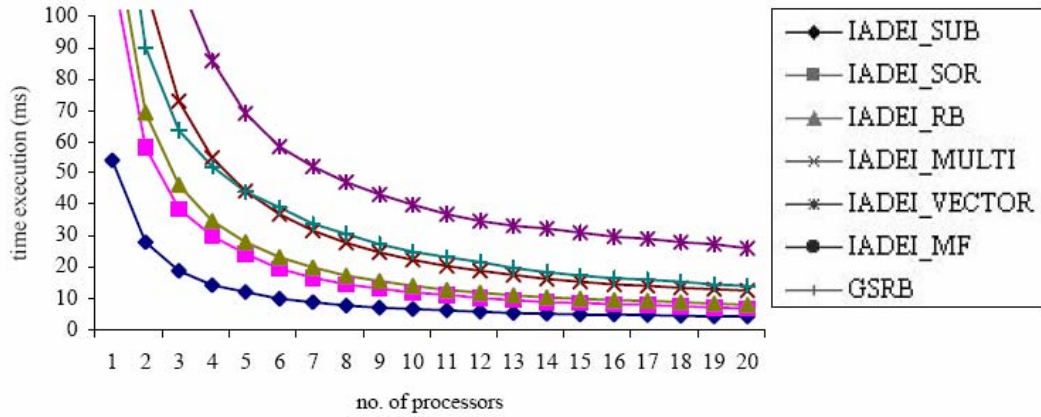


Figure 4.2 : The execution time vs. number of processors

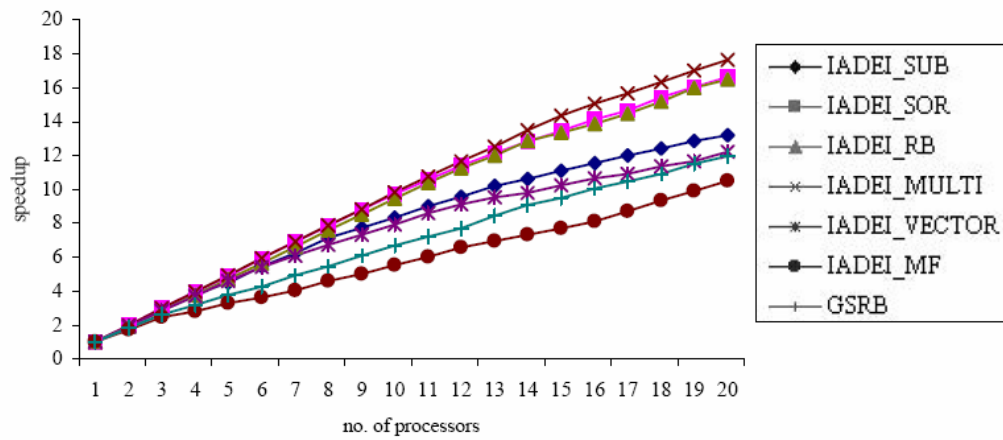


Figure 4.3 : The speedup vs. number of processors

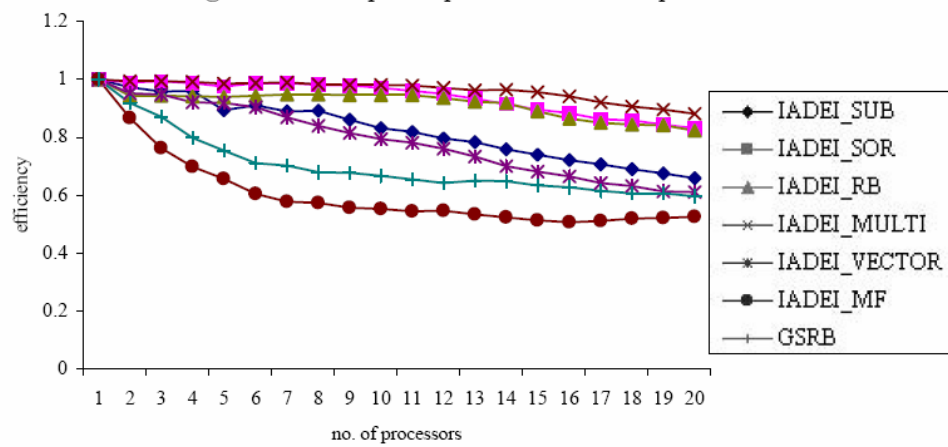


Figure 4.4 : The efficiency vs. number of processors

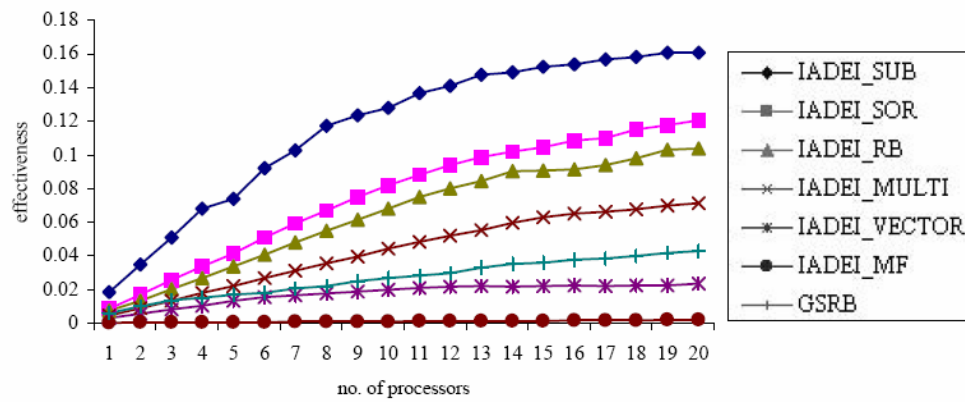


Figure 4.5 : The effectiveness vs. number of processors

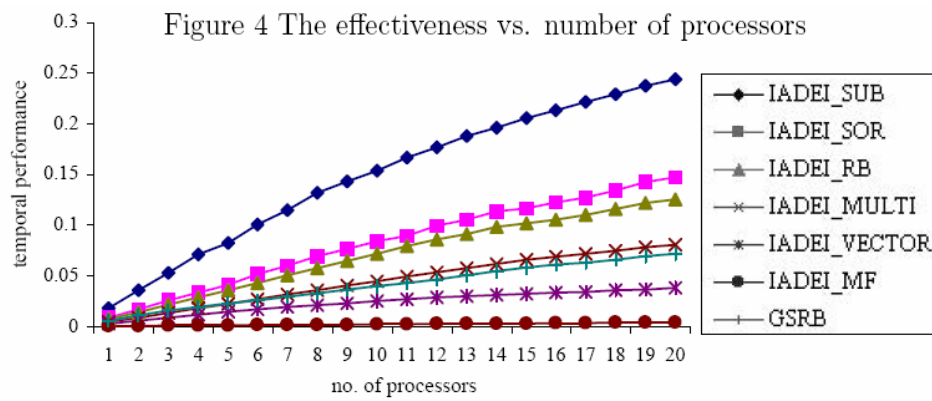


Figure 4.6 : Temporal performance vs. number of processors

Table 2 Computational complexity per iteration for the parallel strategies of IADEI methods

method	c.computation	c.communication
SUB	$(\frac{2408m}{p} + 4214)T + (\frac{3010m}{p} + 4816)D$	$3010t_{data} + 2408(t_{start} + t_{idle})$
SOR	$\frac{3580mT}{p} + \frac{5012mD}{p}$	$5012t_{data} + 2864(t_{start} + t_{idle})$
RB	$\frac{3600mT}{p} + \frac{5040mD}{p}$	$5040t_{data} + 2880(t_{start} + t_{idle})$
MULTI	$\frac{3600mT}{p} + \frac{4680mD}{p}$	$5762t_{data} + 3600(t_{start} + t_{idle})$
VECTOR	$\frac{2112012mT}{p} + \frac{2652804mD}{p}$	$2113584t_{data} + 1057050(t_{start} + t_{idle})$
MF	$261(\frac{pD}{\sum_{i=1}^p i} + 2m - \frac{3m}{p} + \sum_{i=1}^p \frac{pT}{i} + 14p - 14)$	$3136t_{data} + 2610(t_{start} + t_{idle})$
GSRB	$(\frac{1800m}{p} + 1200)T + (\frac{2400m}{p} + 1800)D$	$8400t_{data} + 4800(t_{start} + t_{idle})$

c.computational=computational complexity , c.communication= communications cos, D=multiplications, T=additions

4.3 AGE Methods

The numerical results for sequential BRIAN and DOUGLAS methods are displayed in Table below. This table provides the absolute errors of the numerical solutions using (600x600) and (1000x1000) sizes of matrices. The higher an accuracy of the BRIAN scheme when compare with the DOUGLAS and GSRB methods are evident from these errors. It is reflected from the lower magnitude of the root mean square error (rmse).

The rate of convergence of BRIAN is better than DOUGLAS and GS- RB methods. the computational complexity and communication cost for parallel BRIAN is lower than DOUGLAS and GSRB methods. Performance measurements for the execution time, speedup, efficiency, effectiveness and the temporal performance versus number of the processors was plotted. The parallel execution time and computation time are decreasing with the increasing p. Compared to BRIAN, DOUGLAS and GS- RB methods, sending a larger value of messages and the frequency of communications are reflected the communication time. The communication time of BRIAN method is lower than

DOUGLAS and GSRB methods. The increasing of idle time is due to several factors such as network load, delay and load imbalance.

The reductions in execution time often becomes smaller when a large number of processors is used. A nice speedup can be obtained for all applications with 20 processors. The efficiency of the DOUGLAS method decreases Faster than BRIAN method. The BRIAN method is good in terms of effectiveness and the temporal performance where data decomposition is run asynchronously and concurrently at every time step with the limited communication cost. As the result, the BRIAN method allows inconsistencies due to load balancing when the extra computation cost is needed for boundary condition.

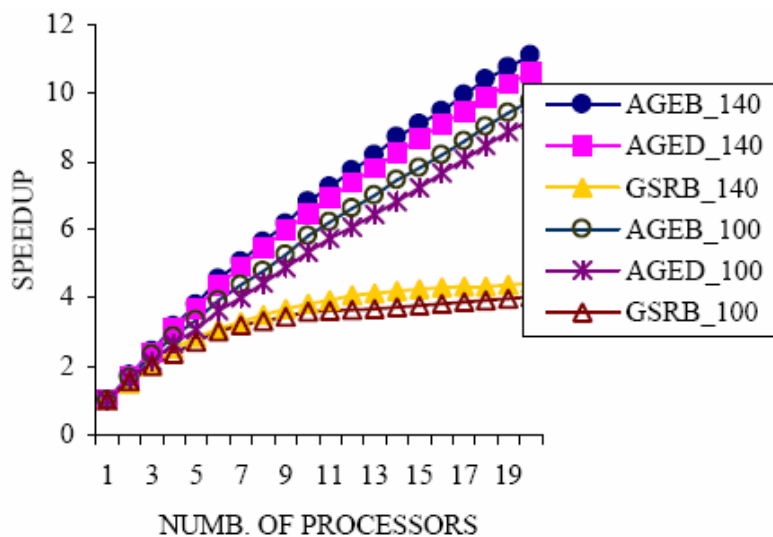


Figure 4.7 : The speedup vs. number of processors

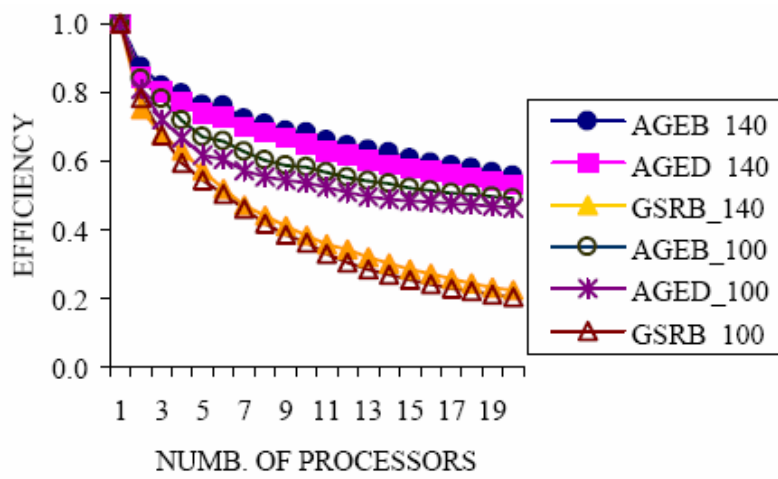


Figure 4.8 : The efficiency vs. number of processors

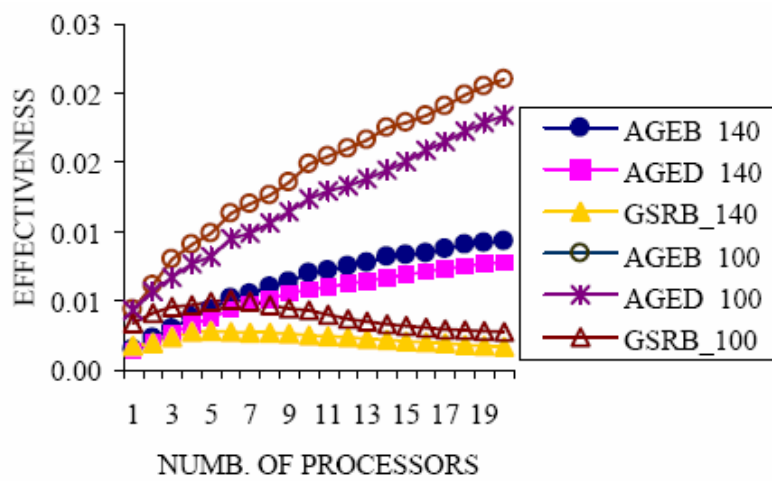


Figure 4.9 : The effectiveness vs. number of processors

Table 3 : Performance measurements of the sequential BRIAN and DOUGLAS methods

$$m = (600 \times 600) - \Delta x = 1.67E^{-3}, \Delta y = 1.67E^{-3}, \Delta t = 1E^{-6}, \lambda = 3.58E^{-1}, \theta = 1.00, \epsilon = 1.0E^{-12}$$

$$m = (1000 \times 1000) - \Delta x = 1.00E^{-3}, \Delta y = 1.00E^{-3}, \Delta t = 1E^{-6}, \lambda = 1, \theta = 1.00, \epsilon = 1.0E^{-12}$$

m	600 × 600			10 ³ × 10 ³		
AGE	BRIAN	DOUGLAS	GSRB	BRIAN	DOUGLAS	GSRB
$t_{para}(\mu s)$	33.8854	35.6894	40.1592	138.9811	143.2267	157.6732
iterations	130	150	500	200	230	622
rmse	$3.07166E^{-12}$	$3.07398E^{-12}$	$3.07461E^{-12}$	$2.4396E^{-12}$	$2.4416E^{-12}$	$2.4482E^{-12}$
r	$6.6968E^{-13}$	$6.6302E^{-13}$	$6.6303E^{-13}$	$7.3807E^{-13}$	$7.3185E^{-13}$	$7.3988E^{-13}$
ave_rmse	$1.0120E^{-11}$	$1.0116E^{-11}$	$1.0151E^{-11}$	$8.0722E^{-12}$	$8.0597E^{-12}$	$8.0979E^{-12}$
r.maks	$1.5646E^{-23}$	$1.5668E^{-23}$	$1.56811E^{-23}$	$9.8766E^{-24}$	$9.8938E^{-24}$	$9.9535E^{-24}$
r	0.99	0.95	—	1.6	1.6	—

rmse= root means square error, |r|= absolute error, r.maks = maximum error and ave_rmse = average of rmse

Table 4 : Computational complexity for parallel BRIAN and DOUGLAS methods

m	600 × 600		10 ³ × 10 ³	
kaedah	add.	mult.	add.	mult.
coefficients AGE	7	4	7	4
BRIAN	$\frac{3250m^2 - 3380m}{p}$	$\frac{2680m^2 - 2600m}{p}$	$\frac{5000m^2 - 5200m}{p}$	$\frac{4400m^2 - 4000m}{p}$
DOUGLAS	$\frac{3750m^2 - 3900m}{p}$	$\frac{4500m^2 - 3900m}{p}$	$\frac{5750m^2 - 5980m}{p}$	$\frac{5060m^2 - 4600m}{p}$
GSRB	$\frac{5000m^2 - 5000m}{p}$	$\frac{5500m^2 - 5000m}{p}$	$\frac{6200m^2 - 6220m}{p}$	$\frac{6842m^2 - 6220m}{p}$

add.= additions, mult.= multiplications and p = number of processors

Table 5: Communication cost for parallel BRIAN and DOUGLAS methods

m method	600×600 communication cost	$10^3 \times 10^3$ communication cost
coefficients AGE	0	0
BRIAN	$520mt_{data}$ $+520(t_{startup} + t_{idle})$	$800mt_{data}$ $+800(t_{startup} + t_{idle})$
DOUGLAS	$600mt_{data}$ $+600(t_{startup} + t_{idle})$	$920mt_{data}$ $+920(t_{startup} + t_{idle})$
GSRB	$2000mt_{data}$ $+2000(t_{startup} + t_{idle})$	$2488mt_{data}$ $+2488(t_{startup} + t_{idle})$

Table 6 : Execution, communication and idle times (μ sec) for parallel BRIAN and DOUGLAS methods

m	600X600					1000X1000				
method	BRIAN					BRIAN				
p	para	comp	comm	comm1	idle	para	comp	comm	comm1	idle
5	8.7500	6.3771	2.3729	0.7930	1.5799	30.480	27.396	3.0838	2.0200	1.0638
%		72.9	27.1	9.1	18.1		89.9	10.1	6.6	3.5
10	5.1220	3.1885	1.9335	0.7930	1.1405	17.433	13.698	3.7353	2.0200	1.7153
%		62.3	37.7	15.5	22.3		78.6	21.4	11.6	9.8
15	3.7600	2.1257	1.6343	0.7930	0.8413	12.788	9.1321	3.6557	2.0200	1.6357
%		56.5	43.5	21.1	22.4		71.4	28.6	15.8	12.8
20	3.0670	1.5943	1.4727	0.7930	0.6797	10.659	6.9491	3.7100	2.0200	1.6900
%		52.0	48.0	25.9	22.2		65.2	34.8	19.0	15.9
method	DOUGLAS					DOUGLAS				
p	para	comp	comm	comm1	idle	para	comp	comm	comm1	idle
5	9.9971	6.7379	3.2592	0.9150	2.3442	34.250	28.245	6.0047	2.3230	3.6817
%		67.4	32.6	9.2	23.4		82.5	17.5	6.8	10.7
10	5.8260	3.3689	2.4571	0.9150	1.5421	18.955	14.123	4.8323	2.3230	2.5093
%		57.8	42.2	15.7	26.5		74.5	25.5	12.3	13.2
15	4.4507	2.2460	2.2047	0.9150	1.2897	13.990	9.4151	4.5749	2.3230	2.2519
%		50.5	49.5	20.6	29.0		67.3	32.7	16.6	16.1
20	3.6070	1.6845	1.9225	0.9150	1.0075	11.798	7.0613	4.7367	2.3230	2.4137
%		46.7	53.3	25.4	27.9		59.9	40.1	19.7	20.5
method	GSRB					GSRB				
p	para	comp	comm	comm1	idle	para	comp	comm	comm1	idle
5	14.006	7.6318	6.3742	2.4400	3.9342	42.302	31.135	11.167	5.0258	6.1416
%		54.5	45.5	17.4	28.1		73.6	26.4	11.9	14.5
10	8.2012	3.8159	4.3853	2.4400	1.9453	23.656	15.567	8.0887	5.0258	3.0629
%		46.5	53.5	29.8	23.7		65.8	34.2	21.2	12.9
15	6.0150	2.5440	3.4711	2.4400	1.0311	17.300	10.378	6.9218	5.0258	1.8960
%		42.3	57.7	40.6	17.1		60.0	40.0	29.1	11.0
20	5.1231	2.0080	3.1151	2.4400	0.6751	14.524	8.1000	6.4240	5.0258	1.3982
%		39.2	60.8	47.6	13.2		55.8	44.2	34.6	9.6

para= t_{para} , comp= t_{comp} , comm= t_{comm} , idle= t_{idle} , comm1= $\alpha t_{data} + \beta t_{start}$ with α and β dependent on m and L .
 % = the gain of t_{para} .

Table 7 : Computational / communication ratio for the parallel BRIAN and DOUGLAS methods

method	m / p	4	8	16	20
BRIAN	1000×1000	8.8840	3.6672	2.4980	1.8731
	600×600	2.6874	1.6491	1.3007	1.0825
DOUGLAS	1000×1000	4.7039	2.9225	2.0580	1.4908
	600×600	2.0673	1.3711	1.0187	0.8762
GSRB	1000×1000	2.7880	1.9246	1.4994	1.2609
	600×600	1.1973	0.8702	0.7329	0.6446

p = number of processors

Table 8 : Performance measurements vs. number of processors for the parallel BRIAN and DOUGLAS methods

m Perform.	600 BRIAN	× DOUG.	600 GSRB	1000 BRIAN	× DOUG.	1000 GSRB
T_p						
4	10.598	11.875	16.4015	37.55001	41.15	51.819
8	5.9989	6.8671	9.7508	20.58555	22.81	28.289
12	4.4598	5.2051	7.0302	15.10335	16.535	20.442
16	3.5710	4.2514	5.7922	12.19765	13.35	16.7423
20	3.067	3.607	5.1231	10.65897	11.7980	14.5235
S_p						
4	3.6203	3.4390	2.96448	3.86557	3.64945	3.28577
8	6.3958	5.9472	4.98648	7.05116	6.58374	6.01880
12	8.60317	7.84615	6.91616	9.61058	9.0822	8.32908
16	10.7444	9.6064	8.39439	11.90000	11.2490	10.1698
20	12.5101	11.3224	9.49073	13.61783	12.7288	11.7234
C_p						
4	0.90508	0.8597	0.74112	0.96639	0.91236	0.8214
8	0.79948	0.7434	0.62331	0.88139	0.82297	0.75235
12	0.71693	0.65385	0.57635	0.80088	0.75686	0.69409
16	0.67153	0.6004	0.52465	0.74375	0.70307	0.63561
20	0.6255	0.56612	0.47454	0.68089	0.63644	0.58617
L_p						
4	0.09436	0.08421	0.06097	0.02663	0.02430	0.01929
8	0.1667	0.14562	0.10256	0.04858	0.04384	0.035349
12	0.22423	0.19212	0.14224	0.06621	0.06048	0.048917
16	0.28003	0.23522	0.17265	0.08198	0.07491	0.05972
20	0.32605	0.27724	0.19519	0.09382	0.08476	0.06885

S_p = speedup, C_p =efficiency, F_p = effectiveness, L_p =temporal performance, T_p = parallel execution time

Chapter 5

Concluding Remarks and Suggestions

5.1 IADE Methods

A new approach for study of the IADE method using linear interpolation concept. A method of this kind, namely IADEI was discussed in solving one dimensional parabolic partial equations. A comparison with the parallel strategies of IADEI scheme shows that the IADEI-SUB has extended range of efficiency, speedup and effectiveness. Furthermore, the superiority of the IADEI-SUB is also indicated by the highest value of the temporal performance, accuracy and convergence in solving large-scale linear algebraic equations on a distributed memory multiprocessors platform, which is superior for all numbers of processors. As the conclusions:

1. Some parallel strategies code for PDE problems has been successfully developed and validated.
2. The method of parallelization by both the method of IADE and IADEI parallelization using domain decomposition technique were successfully implemented into SIMD codes.
3. The convergence rates are found to be independent of number of partitions and iterations.
4. The sequential of IADEI method has lower speedups and poorer scalability than the parallel IADEI-SUB method, due to rapid rise in the time spent on communication as a result of denser coarse grids used.

5.2 AGE Methods

The computational complexity of the algorithms, which clearly shows that the BRIAN method has less memory accesses than DOUGLAS method with the limited

communication time. The result on a cluster of workstations shows that BRIAN method is a more efficient algorithm than DOUGLAS. BRIAN method is shown to be more accurate than the corresponding DOUGLAS scheme. Parallel BRIAN method is inherently explicit, the domain decomposition strategy is efficiently utilized, straightforward to implement on a cluster of workstations.

Therefore, we reach the conclusion that communication and computing times are affected the speedup ratio, efficiency and effectiveness of the parallel algorithms. Sending a larger value of messages and the frequency of communications are reflected in the communication time.

6. REFERENCES

<http://en.wikipedia.org/wiki/Parallel%5Fcomputing> Wednesday February 2, 2006

Alias, N., Sahimi, M.S., Abdullah, A.R., (2004). The development and Implementation of Efficient Parallel Strategies on a Cluster Sistem Using a Message-Passing Applications. Proceeding of 7 th International Conference On Work With Computing Sistems. 869-873.

Alias, N., Yan C. H. Arriffin A.K. & Sahimi, M.S., (2004). High Performance Computing for 2 Space Dimension Problems on PC Cluster Sistem. Proceeding of 7 th International Conference On Scientific And Engineering Computation. Singapore. CD Format.

Alias, N., Sahimi, M.S., Abdullah, A.R., (2004). PVM-based implementation of and ADI in solving one dimension parabolic Equation using PC Cluster Sistem. International Conference On Mathematics and Its Applications. ICMA, Kuwait University. Pg 5.

Alias, N., Sahimi, M.S., Abdullah, A.R., (2004). Communication issues for Parallel Alternating Group Explicit Methods on a Distributed Memory Architecture. 2nd International Conference On Artificial Intelligence in Engineering and Technology. Pg 840-844.

Chalmer, A. Tidmus, J. (1996). Practical Parallel Processing An Introduction to Problem Solving in Parallel. International Thomson Komputer Press.

Flynn, M.J. (1972). Some komputer organizations and their effectiveness. IEEE Trans. On Komputers. 21 (9): 948- 960.

Quinn, Michael J. (2004) Parallel Programming in C with MPI and OpenMP
McGrawHill