INCOMPRESSIBLE FLOW SIMULATION USING S.I.M.P.L.E METHOD ON PARALLEL COMPUTER

BUKHARI BIN MANSHOOR

A dissertation submitted in partial fulfilment
of the requirement for the award of the degree of
Master of Engineering (Mechanical-Pure)

Faculty of Mechanical Engineering
University of Technology Malaysia

DECEMBER 2005

To my beloved family,

The love in you brings my dream comes true.

To my baby, who has brought a new level of love, patience

and understanding into our lives.

ACKNOWLEDGEMENT

"In the name of Allah that the most Gracious, the most Merciful"

We can dream, create, design and even develop the world class technology, but we need blessing and assistance from The Sustainers to make it comes true. Praise be to Allah, The Cherisher and Sustainers of the world; Most Gracious; most Merciful; Master of the Day of Judgment.

First foremost, all praise and thanks goes to Allah s.w.t who gave me the strength and blessings to make it possible to complete this thesis titled **Incompressible Flow Simulation using S.I.M.P.L.E Method on Parallel Computer.**

Secondly, the author would like to convey his deepest sincere gratitude to the advisor and co-advisor of this project, Prof. Dr. Mat Nawi Wan Hassan and Dr. Hjjh. Norma binti Alias, for all their support, guidance, advice and motivation and also for their confidence in my ability to succeed.

The author also would like to take this opportunity to extend his never-ending gratitude towards his family for their continuous support and prayers that make it possible for the author to succeed. In addition, thank you to all lecturers, UTM staffs, friends, organisations and individually who directly or indirectly have helped to make this Master project possible.

Thank you again to each and everyone who have in any way contributed to this Master project. May Allah forgive us and bless us all. Ameen.

ABSTRAK

Komputer selari merupakan gabungan beberapa pemproses yang bertujuan meningkatkan keupayaan sesebuah sistem komputer dalam melaksanakan sesuatu pengatucaraan. Dalam projek ini, sistem komputer selari yang digunakan dikenali sebagai sistem komputer selari berkelompok. Kelebihan menggunakan sistem komputer selari berkelompok ini ialah ia mampu bergerak sendiri sebagai komputer sesiri jika tidak beroperasi sebagai komputer selari. Perisian komputer selari yang boleh digunakan sebagai sistem operasi kepada sistem komputer selari berkelompok ini termasuklah UNIX, Window NT atau Linux. Projek ini memberikan penumpuan dalam penggunaan sistem komputer selari berkelompok menggunakan perisian PVM untuk menyelesaikan persamaan Navier-Stoke dalam membuat simulasi dua dimensi aliran tidak boleh mampat dalam ruang segiempat. Kaedah yang digunakan adalah berasaskan algoritma SIMPLE dan algoritma SIMPLE yang telah diubahsuai dengan menggunakan kaedah pembahagian domain dan kaedah pembahagian fungsi. Ketepatan kedua-dua kaedah tersebut telah dibandingkan dengan keputusan piawai yang berkaitan dengan masalah aliran dua dimensi dalam ruang segiempat. Keupayaan kedua-dua kaedah tersebut dari segi masa perlaksanaan, kecepatan dan keberkesanan juga telah diperolehi dan didapati penggunaan komputer selari telah memberikan prestasi yang lebih baik dalam menyelesaikan masalah persamaan Navier-Stoke tersebut. Dengan kaedah pembahagian domain, didapati masa perlaksanaan dapat dikurangkan sebanyak 70% manakala dengan menggunakan kaedah pembahagian fungsi, masa perlaksanaan dapat dikurangkan sebanyak 25 % berbanding dengan menggunakan komputer sesiri.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

E	-	efficiency
g	-	gravitational acceleration
H	-	height of cavity
L	-	length of cavity
$K(p)$	-	effectiveness
Nu	-	Nusselt number of cavity $-(H/\Delta T)(\partial T/\partial x)_w$
$\overline{Nu}$	-	Averaged Nusselt number in cavity $\int_0^1 Nu\, d(y/H)$
p	-	pressure
Pr	-	Prandtl number
Ra	-	Rayleigh number $g\beta\Delta TH^3\Pr/v^2$
$R(p)$	-	Impermanent performance
S	-	Gradient of thermal stratification in cavity centre $(H/\Delta T)/(\partial T/\partial y)$
$S(p)$	-	Speed-up factor
T	-	temperature
ΔT	-	characteristic temperature difference for the cavity $T_h - T_c$
T_h	-	temperature of the hot cavity wall
T_c	-	temperature of the cold cavity wall

t_p	-	total execution time in parallel
t_{seq}	-	total execution time in sequential
u	-	horizontal velocity component
v	-	vertical velocity component
x	-	horizontal coordinate
y	-	vertical coordinate

Greek symbols

β	-	coefficient of thermal expansion
ν	-	molecular kinematic viscosity
ρ	-	density
ψ	-	stream function $u = -\partial\psi/\partial y \quad v = \partial\psi/\partial x$
ψ_c	-	stream function at centre of cavity

Subscripts

max	-	maximum of the quantity
w	-	wall condition
∞	-	environmented condition

CHAPTER 1

INTRODUCTION

1.1 Background

Computational fluid dynamics (CFD) has advanced rapidly over the last two decades and it is recognized as a valuable tool for engineering applications. However, numerical simulation of viscous flow fields remains very expensive even with the use of current vector computers. Advances in numerical algorithms are not expected to reduce the cost of those computations to the extent that they can routinely be applied for design.

Vector computers consist of a few powerful processing units that work independently accelerating computations by one or two orders of magnitude compared to scalar machines, which are not sufficient for efficient large scale flow simulations. Another approach to computer architectures is the employment of a number of processors that work in parallel executing the same job. Parallel computing appears to be a promising approach for future design application of CFD.

Parallel computing or also known as parallel processing refers to the concept of speeding up the excitation of a program using multiple processors by dividing the program into multiple fragments that can execute simultaneously, each on its own processor. A program being executed across *n* processors might execute *n* times faster that it would use a single processor. Parallel processing differs from multitasking in which single processors execute several programs at once.

Since a new generation of single processor computer is a costly enterprise in order to obtain a larger and faster communications, parallel computing becomes a key for high performance architecture. All cotemporary supercomputers are parallel processing computers. Massively parallel processors (MPPs) are now the most powerful computer in the world. These machines combine a few hundred to a few thousand CPUs in a single large cabinet connected to hundreds of gigabytes of memory.

There are two methods of parallel processing.

i. Domain Decomposition Method – Domain decomposition or "data parallelism", data are divided into pieces of approximately the same size and then mapped to different processors. Each processor then works only on the portion of the data that is assigned to it. Of course, the processes may need to communicate periodically in order to exchange data. Data parallelism provides the advantage of maintaining a single flow of control. A data parallel algorithm consists of a sequence of elementary instructions applied to the data an instruction is initiated only if the previous instruction is ended.

ii. Functional Decomposition Method - Functional decomposition or also known as task parallelism, the problem is decomposed into a large number of smaller tasks and then, the tasks are assigned to the processors as they become available. Processors that finish quickly are simply assigned more work. Task parallelism is implemented in a client-server paradigm.

This project deals with the solving of an incompressible flow simulation using Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) that originally put form ward by Patankar and Spalding (1972). As we know, the analysis of an incompressible flow become more complicated and need a high performance computer to solve the problem. One of the problems in solving the complicated problem of an incompressible flow is the time constraint. More complicated of the problem means more time should be spend to solve the problem.

To overcome this problem, parallel computer was used and to determine the performance of this parallel computations, the corresponding parallel algorithms was developed and it based on the two methods of parallelization there are Domain Decompositions Method (DDM) and Functional Decomposition Method (FDM). At the end of this project, the result for both simulations using parallel algorithms are presented and discussed.

1.2 Objective of the Project

The objective of this project is to develop a code of parallel algorithm and to determine the performance of the code on parallel computer. The numerical procedure are based on modified SIMPLE algorithm and the parallelization methods used both Domain Decomposition Technique and Functional Decomposition Method.

The model used to analyse performance of the parallel computations is the problem of natural convection occurred in a square cavity with specified boundary conditions. The flow in a square cavity that is considered here is due to natural convection. The fluid used is air with a Prandtl number of 0.71. The aspect ratio L/H is 1.

The flow is described by the Navier-Stokes equations under the Boussinesq approximation that will discuss later. The summary of the boundary conditions chosen are as follow:

i. Both the upper and the lower wall are adiabatic.
ii. The vertical walls are isothermal; the left wall is at hot temperature T_h and the right wall is at cold temperature T_c.
iii. Velocities at all boundaries are zero.

The model of square cavity was shown in Figure 1.1.

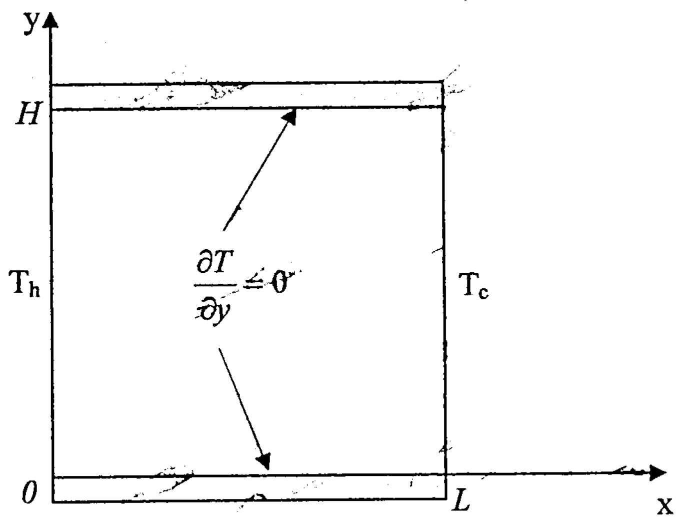

Figure 1.1 Model of a square cavity.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Since the parallel computations were introduced, research on incompressible flow is expanding towards more on applications of solving engineering problems. This is because using parallel computations, the problems of time constraint can be reduced. It means that the complicated problems can be solved with more faster than the serial computer. Starting early 90's, many researchers tried to solve a past problem of fluid dynamics using parallel computers. They reworked on the past problem with a little modification using parallel computer. They found that the problems can be solved faster and easier than before, even though they have to use complicated algorithms.

A.W Date (1986) modified a SIMPLE algorithm by adding an extra pressure correction step on the original SIMPLE algorithms. The purpose of the modification is to accelerate the convergence of pressure relaxation. For the purpose of study, the author tried to solve a natural convection heat transfer in horizontal annulus using two modifications on SIMPLE algorithms. As a result, the SIMPLE algorithm was found to be extremely slow to converge and even after 1000 iterations the heat balance only 2.7%. By using the modification of SIMPLE algorithm on parallel processing, it showed that the result nearly 4 times faster although it involved solution of a greater number of equation per iteration.

In order to give a faster rate of convergence of the solution of the pressure-correction equation, V.A.O. Anjorin and I.E. Barton (2001) modified a SIMPLE algorithm and developed SIMPLEV (SIMPLE-Vincent). A proposed improvement is made to enhance the convergence rate of this algorithm. This is aimed at giving a faster rate of convergence of the solution of the pressure-correction equation (Versteeg and Malalasekera, 1995) than the SIMPLE algorithm. The SIMPLEV algorithm uses the same methodology as that of the SIMPLE algorithm in solving the velocity fields that satisfy the continuity equation except that the under relaxation and temporal terms are removed from the pressure-correction equation of the SIMPLE algorithm. When this is performed the pressure correction tends to zero therefore satisfying the continuity equation to obtain better convergence. The problem on this algorithm is for grid systems with large number of nodes, the efficiency of the SIMPLEV algorithm will reduces.

With the interesting of using SIMPLE method on solving an incompressible Navier-Stokes equation, G.Horton (1992) tried to solve an incompressible Navier-Stokes equation using SIMPLE with time-parallel method. According to his study, he found that the advantage of time-parallel approach over standard parallel methods is the retainment of larger vector lengths. Grid portioning schemes suffer from the reduction of vector length incurred by the subdivision of the computations grid. However, several problems often associated with the implementation of space-parallel methods such as load balancing, restructuring of sequential code and the mapping of the problem onto the parallel machine.

Shinsuke Kato, Shuzo Murakami, Wei Zhang, Nabuhiro Miura, Tadashi Okamoto (1995) also studied on incompressible flow simulation on parallel computer. However, there were used a SIMPLE algorithm that was modified by Date (1986) and known as SIMPLE-D method. The computer used in this study is ADENART (Alternating Direction Edition Nexus ARray system). In their study, they were compared the analysis between two system there are ADENART64 (with 64 processor) and ADENART256 (with 256 processor). 2D and 3D square cavity was simulated with ADENART64 and ADENART256 and from the result that was obtained, the ADENART256 showed that it have a lower calculation time (in second) compare with ADENART64 for the calculation time of 100 iteration, 82 x 82 grid system, calculation time for ADENART64 is 74.5 second while ADENART256 only take 37.9 second.

When the parallel computer becomes widely used in solving a fluid flow problem, groups of researcher start looking on the method of parallelization. Kenjiro Shimano and Chuichi Arakawa (1995) had tried to solve an incompressible flow simulation using parallel computations with domain decomposition technique. During their study, they try to examine the effect of domain decomposition technique on the convergence property and also to determine how frequently processor should communicate to obtain the best efficiency. To achieve this propose, the authors were calculated the laminar flow in a square cavity for laminar flow and the turbulent flow in a suddenly expanding pipe. From their study, they were found that the parallelization efficiency will increase as the number of grid increase. It showed that the domain decomposition technique on parallelization will give the best performance during to solve the incompressible flow.

Other groups of researcher that try to vary the methods of parallelization are Jerome Breil, Rossitza S. Marinova, Hideiki Aiso and Tadayasu Takahashi (2003). They were presented a fully coupled method to solve incompressible Navier-Stokes equations. The parallelization of their implicit method is based on domain decomposition method. As a result, the scheme that was produced has second order accuracy and the domain decomposition allows solving a big problem in less computational times. The speed-up will increase by using multy-dimensional domain decomposition methods.

Based on the research that had been done, one of the interesting about the parallel computer is about the convergence properties. According to this interesting, San-Yih Lin and Zhong-Xin Yu (2003) compared the various numerical method for incompressible Navier-Stokes Equations. To analyze the convergent rate, the authors were used an explicit Runge-Kutta and implicit DDADI method and compared in the parallel computations. For complex configurations computation, the multizov\ne technique is usually utilized to generate a grid system. The result showed that the

DDADI methods performs better that the Runge-Kutta method does. In their paper work also showed the multizone division will affect the convergent rate. However, the performance of the DDADI method in the parallel computations still efficiency for solving the incompressible flow problem.

2.2 Overview of Parallel Programming

For over 40 years, virtually all computers have followed a model known as the von Neumann computer. This model was introduced by Hungarian mathematician, John von Neumann. A von Neumann computer uses the stored-program concept. The CPU executes a stored program that specifies a sequence of read and writes operations on the memory. This model absolutely differs with a parallel computer where in the parallel computer, a collection of processors are used to speed up the execution of one program.

The history of the growth of parallel programming is showed in Table 2.1 below.

Table 2.1: The history of parallel computer architecture development.

Year	Parallel Computer	Architecture
1976	Cray - 1	First pipelined
1981	BBN Butterfly	256 processor Motorola 68000
1982	Cray X/MP	4 processors of Cray-1: Shared memory
1986	TMC CM – 1	64k processors 1-bit, 12-D hypercube network
1988	Intel iPSC/2	Intel processor 80386, 7-D hypercube network
1989	Fujitsu VP2000	2 multiprocessors processors
1991	KSR-1	32 processors shared virtual memory machine
1992	TMC CM-5	1204 processors SPARC with tree topology
1995	Intel P6	9000 processors machine based on P6

The classification of the parallel computer architecture can be divided into three categories: Flynn's taxonomy, Quinn classification and Cheong classification.

2.2.1 Flynn's Taxonomy

Flynn's Taxonomy is one of the standard ways of classifying computer systems in that proposed by Flynn (1972). Flynn's Taxonomy distinguishes multi-processor computer architectures according to how they can be classified along two independent dimensions of instruction and data. Each of these dimensions can have only of two possible states, single or multiple. According to this classification there are four basic types there are;

i. Single Instruction – Single Data (SISD)
   The Von Neumann model
ii. Single Instruction – Multiple Data (SIMD)
   These include machines supporting array parallelism
iii. Multiple Instruction – Single Data (MISD)
   No systems have been built which fit this classification
iv. Multiple Instruction – Multiple Data (MIMD).
   Covers the multiprocessors systems supporting process parallelism on which we concentrate

2.2.2 Quinn Classification

Quinn (1994) verified that the architecture of parallel computer can be classified according to the memory architecture, organization of processors, number of processors that communicate between each other and the flow of the data. Memory and processors organization can be divided into a parallel computer system and distributed parallel computer system.

Parallel computer system consists of two or more unit of similar processors, special and limited, organization and architecture of the system (Huang & Douglas, 1989). Processors that build in this system will connect by a networking with the fastest cross communication. Every processor shars the data and executes a job simultaneously among each processor.

The distributed parallel computer system consists of several processors units that not related for each other and will connected with a networking to do a sending message process for communication between the processors. Every processor working autonomously and consist of their own local memory. According to the memory organization class, this parallel computer system will be categorized as multiprocessors computer system and multicomputer computer system.

2.2.3 Cheong Classification

According to Cheong (1992), parallel computer system will be categorized into the three main structures. The structures are as follow;

i. Array multiprocessor computer
   Array multiprocessor computer is a machine consisting of a processor that operating with parallel on the different input. This type of structure has a very high speed and efficiency. The example of this parallel computer is a Cray Y/MP.

ii. Combination of parallel processor
   Combination of parallel processor consists of a set of processor that is bigger than a set of array multiprocessor computer. A classification of the design system for combination parallel processor is the same like Taxonomi Flynn classification.

iii. Cluster of workstations

Cluster of workstations is a new feature on a parallel computer system design. The cluster of workstations is a structure of multi workstation that rapidly growth. This is because the cost of building the parallel application with big memory storage is cheaper compare with multiprocessor parallel computer. At the same time, achievement of this cluster of workstations is approaching the multiprocessor parallel computer (Phyllis & Clement, 1996).

Development of the cluster of workstations only needs a several workstations or personal computers that connected with a low cost network such as Ethernet with Linux configuration (Grammatikakis, 2001). Domain software packages for communication system such as PVM (Geist et al. 1994a & 1994b), MPI (Gropp, 1999) or PARMACS can be obtained freely from internet.

CHAPTER 3

BACKGROUND THEORY

3.1 Mathematical Formulation

The equations governing the fluid dynamics and energy flow have been known for the most part for more than a century and yet have continued to defy analytical solution. Instead their solutions have largely been obtained by experimental simulations in wind tunnels, water tables and shock tubes. Now with the availability of advanced scientific computers, the equations can be solved using the methods of computational fluid dynamics (CFD). With a new trend in using a parallel computer to solve an engineering problem, it is not surprising that, fluid dynamics and heat transfer are contributing to and benefiting from the current development in finite difference numerical analysis.

The numerical solution to a problem related to fluid and energy flows depends on:

i. The number of equations that governs the system undertaken.
ii. The form of the partial differential equations involved - parabolic, hyperbolic or elliptic.
iii. The linearity of the equations.
iv. The system of the equations - coupling or not.
v. The source term in the equations.

REFERENCES

1. A. W. Date (1985). "*Numerical Prediction of Natural Convection Heat Transfer in Horizontal Annulus.*" Int. J. Heat Mass Trasfer.

2. Alias N. (2004). "*Pembinaan dan perlaksanaan algoritma selari bagi kaedah kelas TTHS dan TTKS dalam menyelesaikan persamaan parabolic pada system computer selari ingatan teragih*". PhD Thesis: Universiti Kebangsaan Malaysia.

3. Amdahl, G (1967). "*Validity of the single processor approach to achieving large scale computing capabilities*". Proc. AFIPS Conference. 30: 483-485

4. Buyya, R. (1999). "*High performance cluster computing: Programming and application*". London: Prentice Hall.

5. Cheong, F. C. (1992). "An *agent-oriented programming language for heterogeneous distributed computing*". Thesis PhD. Michigan University.

6. Davis G. de Vahl (1983). "*Natural convection of air in a square cavity: a benchmark numerical solution*". Int. Journal Numerical Meth. Fluid (3): 249-264.

7. Dongarra, J. & Eijkhout, U. (2000). "*Numerical linear algebra algorithms and software*." Journal of Computational and Applied mathematics. 123 (2):489-514.

8. El-Rewini, H., Lewis, T. G. & Ali. H. H (1992). "*Introduction to parallel computing*". New York: Prantice hall.

9. Flynn, M. J. (1972). "*Some computer organizations and their effectiveness*". IEEE Trans. On Computer 21 (9): 948-960

10. Freeman T. L & Philips C. (1992) "*Parallel Numerical Algorithms*". London: Prentice Hall

11. Geist, A. et al. (1994). "*PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*". Massachusetts: The MIT Press.

12. H. K. Versteeg and W. Malalasekera. (1995). "*An Introduction to Computational Fluid Dynamics.*" Pearson, Prentice Hall.

13. Jeremo Breil, Rossitza S. Marinova, Hideaki Aiso and Tadayasu Takahashi (2003). "*Fully Coupled Solver for Incompressible Navier-Stokes Equations using a Domain Decomposition Method.*" Parallel Computational Fluid Dynamics – New Frontiers and Multi-Disciplinary Applications, Edited K. Matsuno et al 249-256.

14. John D. Aderson, Jr. (1995). "*Computational Fluid Dynamics The Basics with Applications.*" International edition. McGraw-Hill Inc, New York.

15. Kenjiro Shimano and Chuichi Arakawa (1995). "*Numerical Simulation of Incopressible Flow on Parallel Computer with the Domain Decomposition Technique.*" Parallel Computational Fluid Dynamics: New Algorithms and Applications, Edited N. Satofuka et al189-196.

16. Quinn, M. J. (1994). "*Parallel computing theory and practice*". London: McGraw Hill.

17. San-Yih Lin and Zhong-Xin Yu (2003). "*Parallel Numerical Method for incompressible Navier-Stokes equations.*" Parallel Computational Fluid Dynamics – New Frontiers and Multi-Disciplinary Applications, Edited K. Matsuno et al 313-320.

18. Shinsuke Kato, Shuzo Murakami, Wei Zhang, Noburo Miura and Tadashi Okamoto (1995). "*Incompressible Flow Simulation using SIMPLE-D Method on HX Network Parallel Computer.*" Parallel Computational Fluid Dynamics: New Algorithms and Applications, Edited N. Satofuka et al177-185.

19. Suhas V. Patankar (1980). "*Numerical Heat Transfer and Fluid Flow.*" McGraw-Hill Inc, New York.

20. T.J. Chung (2002). "*Computational Fluid Dynamics,*" Cambridge University Press.

21. Wilkinson, B. & Allen M. (1999). "*Parallel programming: Techniques and applications using networked workstations and parallel computers*". New Jersey: Prentice Hall.