CANKAYA UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
INFORMATION TECHNOLOGY

MASTER THESIS

EFFICIENT PARALLEL PROCESSING APPROACH BASED ON DISTRIBUTED MEMORY SYSTEMS

IMAD RASHED

JANUARY 2014

Title of the Thesis: **Efficient Parallel Processing Approach Based On Distributed Memory Systems**

Submitted by : **Imad RASHED**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

[illegible]

Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science of Information Technology.

Billur K

Prof. Dr.Billur KAYMAKÇALAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science (M.S) in Information Technology.

[illegible]

Prof. Dr. Taner ALTUNOK
Supervisor

[illegible]

Assist.Prof.Dr. Subhi R. MOHAMMED
Co-Supervisor

Examination Date: 30/01/2014

Examining Committee Members:

Prof.Dr. Taner ALTUNOK (Çankaya Univ.) [illegible]

Assist.Prof.Dr. Subhi R. MOHAMMED(Duhok Polytechnic Univ.) [illegible]

Assoc.Prof.Dr. Hadi Hakan MARAŞ(Çankaya Univ.) [illegible]

Assoc.Prof.Dr. Fahd JARAD(Univ.of Turkish Aeronautical Association) [illegible]

ACKNOWLEDGEMENTS

First of all, praise to GOD **"ALLAH"** on all the blessings, one of these blessings was the help in achieving this research to its end.

I would like to express my sincere gratitude to my supervisor **Prof.Dr. Taner ALTUNOK** and **Assist.Prof.Dr. Subhi R. MOHAMMED** for their valuable guidance, advice and help.

Finally, my special thanks to my family for the endless help, patience and encouragement.

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : IMAD RASHED

Signature : [illegible]

Date :30.01.2014

ABSTRACT

EFFICIENT PARALLEL PROCESSING APPROACH BASED ON DISTRIBUTED MEMORY SYSTEMS

RASHED, Imad
M.S. Department Of Information Technology

Supervisor: Prof. Dr. Taner ALTUNOK
Co-Supervisor: Assist. Prof. Dr. Subhi R. MOHAMMED
January 30 , 50 pages

Complex problems need long time to be solved, with low efficiency and performance. So, to overcome these drawbacks, the approach of breaking the problem into independent parts and treating each part individually.

When treating with problems that need strong processing, it is necessary to break these problems to independent parts and specify each one to a certain processor in such method that these processors can operate in parallel approach. The system that contains such processors will consist of multiple processing units connected via some interconnection network and the software needed to make the processing units work together. These systems can be Shared, Distributed or Hybrid memory systems.

In this thesis, the approach of distributed memory system is depended and based on client/servers principles, the network can contain any number of nodes; one of them is a client and the others are servers. The algorithms used here are capable of calculating the (Started, Consumed, and Terminated) CPU-times, Total execution time and CPU usage of servers and Client hosts. This work addresses an improved approach for problem subdivision and design flexible algorithms to communicate efficiently between client-side and servers-side in the way to overcome the problems of hardware networking components and message passing problems.

Matrix Algebra operations depended as case-study. For this case study, there are many general algorithms and other related algorithms (i.e. Network-Connection-Checking, Load-Division, Massages-Sending/Delivering, Timings-Calculating, Results-Checking, and Results-Receiving/Storing). All these algorithms designed and tested completely by this work. The obtained results are checked and monitored by special programming-checking-subroutines through many testing-iterations and proved a high degree of accuracy. All of these algorithms are implemented using Java Language.

Keywords: Parallel Processing, Clustering , CPU Usage, Load Division , Parallel Computing ,CPU Timing , Distributed Memory System , Client/Servers.

ÖZET

DAĞINIK BELLEK SISTEMLERINDE DAYALI ETKİN PARALEL İŞLEME YAKLAŞIMI

RASHED, Imad
M.S. Bilgi Teknolojileri Bölümü

Tez Yöneticisi: Prof. Dr. Taner ALTUNOK
Ortak Tez Yöneticisi: Yrd. Prof. Dr. Subhi R. MOHAMMED
Ocak 30, 50 Sayfa

Karmaşık sorunları düşük verimlilik ve performansla çözmek için uzun zamana ihtiyaç vardır. Bu sebeple bu zorlukların üstesinden gelmek için, problemleri bağımsız parçalara bölmek ve her birisini ayrı ayrı işlemden geçirme yaklaşımı gerekmektedir.

Problemleri çözmek güçlü bir işlem gerektirdiğinde, bu problemleri bağımsız parçalara ayırıp, bağımsız parçalardan her birisi ile belli bir işlemcinin paralel bir yaklaşımla uğraşması gerekir. Bu işlemcileri kapsayan sistemin, birbirlerine bağlayıcı bir ağ üzerinden bağlanan çoklu işlem birimlerinden ve işlem ünitelerinin birlikte çalışmasını sağlayacak bir yazılımdan oluşması gerekmektedir. Bu sistemler Paylaşılabilir, Dağıtılabilir veya Hibrit hafıza sistemleri olabilir.

Bu tezde, dağıtılmış hafıza sistemine dayanılıyor ve müşterilerin/ server'lerin dayandırılıyor; ağ düğüm rakamlarından her türlü rakamı ihtiva edebilir, onlardan biri müşteri numarası diğerleri ise server'lerin numaralarıdır. Burada kullanılan algoritmalar (Başlamış, Tüketilmiş ve Bitirilmiş)CPU-zaman, toplam icra süresi ve Server ile

Müşterilerin CPU kullanımı hesaplaması yapabilirler. Bu iş, donanım ağı parçaları ve mesaj iletim problemlerinin üstesinden gelmek için problemlerin parçalara ayrılmasına ve müşteri tarafı ve server tarafı için gelişmiş bir yaklaşıma hitap etmektedir.

Bu Bir durum çalışması olarak, Matrix Cebir çalışmalarına dayanmaktadır. Bu durum çalışması için birçok genel olgaritma ve diğer ilgili olgaritmalar vardır (örneğin, ağ-Bağlantı-Kontrol, Yük Bölme, Mesaj Gönderme/ İletme, Zaman- Hesaplama, Sonuçlar-Kontrol ve Sonuçlar- alma/ Depolma) bütün bu olgaritmalar, bu iş ile tamamen tasarlanmış ve test edilmiştir. Alınan sonuçlar, birçok test-yenileme vasıtasıyla özel programlama-kontrol- alt rutinlerle kontrol edilmiş ve izlenmiş ve yüksek derecede doğruluğu ispatlanmıştır. Bütün bu olgaritmalar, Java dili kullanılarak uygulanmaktadır.

Anahtar kelimeler: Paralel İşleme, Kümeleme, CPU Kullanımı, Yük Bölümü, Paralel Bilgisayar, CPU Zamanlama, Dağıtık Bellek Sistemi, Client / Server.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
PP	Parallel Processing
PrC	Parallel Computers
PS	Parallel System
I/O	Input /Output
MP	Multi-Processors
NUMA	Non-Uniform Memory Access
SMP	Symmetric Multiprocessors
PProg	Parallel Programming
IGSCR	Iterative Guided Spectral Class Rejection
OpenMP	Open Multi-Processing
CPU	Central Processing Unit
SISD	Single-Instruction Single-Data
SIMP	Single-Instruction Multiple-Data
MISD	Multiple-Instruction Single-Data
MIMD	Multiple-Instruction Multiple-Data
SPMD	Single Program Multiple Data
DS	Distributed Systems
NOW	Network Of Workstations
SMS	Shared Memory Systems
DMS	Distributed Memory Systems
DM	Distributed Memory
GPU	Graphics Processing Units
MPMD	Multiple Program Multiple Data
LAN	Local Area Network
WAN	Wide Area Network

CHAPTER 1

INTRODUCTION

1.1 Overview

Parallel Processing (PP), which cause to distribute the program into multiple fragments, represents the idea of speeding–up the program execution that can perform simultaneously. In addition, the process of each program is fulfilled by program itself. A program that fulfilled across N processors with comparing to a single processor might execute N times faster [1].

Furthermore, Parallel Computers (PrC) is anticipated to make a hither to unpredictable impact on our lives. Nowadays, parallel computers, which support human in order to approach information via web search engines. Consequently, the searching progresses is like to do typing the keywords. An interconnection network, which permit changing the data among and between the processors, and shared or distributed memories, is required by Parallel computers. Moreover, the processors are connected in a parallel computer system by interconnection networks [2].

The PrC can also be dispersed to control flow and data flow, which are two major categories. Firstly, the principles of control-flow parallel computers are similar to the sequential or von Neumann computer. Secondly, Data-flow parallel computers may relate to a"non-von Neumann" and extremely various, as there is no pointer to activate instruction(s) or a locus of control. The control is totally divided, with the accessibility of operands and triggering the activation of instructions. From here, the control-flow parallel computer will be focused exclusively [3].

Parallel System (PS) consists of multiple processors that communicate with each other using shared memory. Distributed systems, which are systems of computer, consist of multiple processors that connected by a communication network [4].

The provision of the PS must be done with using high-bandwidth that the data supplied to the processors by the ability of input/output devices, and also adequate proportion eliminated by the computation results. The input/output (I/O) capabilities of parallel computers are essential in application of intensive data, which are found in corporate database and network server environments [3]. Furthermore, the high power processors of massive PS for some of applications of data-intensive is not valuable unless the I/O subsystem can protect with the processors [5].

The main argument for using Multi-Processors (MP) is to create powerful computers by simply connecting multi-processors solely. A multi-processor is anticipated to reach faster speed than the fastest single-processor system. In addition, a MP, which is consisting of a number of single processors, is predicted to be more cost-effective than building a high-performance single processor. Another benefit of a MP is fault tolerance. As a result, if a processor fails, the remaining one should be able to provide continued service, albeit with degraded performance [5].

A conventional way in order to increase system’s performance is to utilize MP that can run in parallel to support a given workload. There are two main common MP organizations, which are Symmetric Multiprocessors (SMP) and clusters. More recently, Non-Uniform Memory Access (NUMA) systems have been introduced commercially [6].

An importantly and relatively recent developed computer system design is clustering. Clustering is an alternative computer to SMP as an approach in order to provide high performance and availability and particularly attractive for server applications. Cluster can also be defined as a group of interconnected computers, which leads to work whole computers together simultaneously as a unified computing resource to be able of creating the illusion of being one machine. In addition, the term whole computer means that a system are capable to run on its own, apart from the cluster in the literature computer in a cluster is typically referred to as a node individually [6].

Parallel programming (PProg) and the design of efficient parallel programs, which are computed scientifically for many years, have been set-up in high-performance. The problems of scientific simulation are an important region in the sciences of natural and engineering. More accurately, A massive computing power and memory space are required by simulations or the larger problems of simulations. In the last decades, high-fulfilment researches, which are consisted of new evolvements in part of hardware and software and a progression are steadily evolved in parallel the computing of high-performance, are observed [7].

There has been an opportunity in order to solve a broader range of intensive problems of computation Parallel by increasing the computing power of sequential computers, which are offered by the distributed computing. Important improvements have been achieved in this field in the last 30 years, although there are many unresolved issues so far. These issues arise from various wide areas, like the design of PS and scalable, which interconnects the efficient dispersion of task processes, or growing the parallel algorithms [8].

Client/server computing, which enables the use of low-cost hardware and software, raises local autonomy and data possession, and gives also better performance and more availability. [9].

1.2 Literature survey

Wesley M. Eddy and Mark Allman, [10], 2000, produced an experiment exploring the possibility of using several computers in parallel in order to solve the problems, which take long periods of time to complete on a single machine. Hence, the use of more computers, the total time of calculation can be reduced dramatically. Furthermore, an experiment results showed that for tasks, which involve multiple requests over a long-delay network, adding more machines to the parallel processing system can also help to reduce the negative effects of the network's delay. The researcher has concluded that with more speed a large job can be performed by adding more computers to the task, the role communications time plays in the total execution time, and the impact a long-delay network has on a distributed computing system.

Pan Lei, Et Al, [11], 2002, described three transformations that turn distributed sequential programs into distributed parallel programs. Real-life examples and performance data were presented, and the advantages of their approach were discussed.

Francios Alexandre R.J, [12] 2004, introduced a software architecture model for designing, analyzing, implementing applications, and performing distributed, asynchronous parallel processing of generic data streams. This model provided a universal framework for the distributed implementation of algorithms and their easy integration into complex systems that exhibit desirable software engineering qualities such as efficiency, scalability, extensibility, reusability and interoperability.

Silva, Luciano, Denise Stringhini, Ismar Frango Silveira, [13] 2005, presented a set of design patterns that could be used to model applications that rely on large hierarchical structures, like trees in a parallel and distributed way. They also presented the modeling of several parallel and distributed applications to which the presented set of design patterns are applied.

O.Yaseen Numan, [14] 2010, addressed distributed memory system depends on client/servers principles, the network was consisted of one client and the others were servers. He improved an approach for problem subdivision and design flexible algorithms to communicate efficiently between client-side and servers-side. His algorithms were capable of calculating the several types of timings for the Client and server hosts. Two case studies depended; namely, Matrix Algebra and Sorting Algorithms with Order up to 4096.

Wang, Qing, Zhenzhou Ji, Tao Liu Qing W. [15] 2011, proposed an efficient implementation of parallel programming open multi-processing(OpenMP) on embedded multi-core platform. For embedded multi-core platform, there are limited memory resources. Incorporating limited memory hierarchy into OpenMP was challenging. Their work presented the solution to extend OpenMP custom directive tiling to improve performance. OpenMP with supporting loop tiling is proposed to utilize memory hierarchy. Preliminary performance evaluations showed that performance improvement was gained from the OpenMP implementation on the target embedded multi-core platform.

Fatohi Ban B. , [16] 2011, addressed the building of a software application that consists of three stages; monitoring, controlling and tracking the program that is currently running on a multi-processor systems such as those having (2, 4 and 8) central processing units (CPU). The implementation of these tested-programs represented applying the principles of shared memory system parallel processing as an additional step to her work.

Asaad Farah H. , [17] 2011, built an application algorithm for implementing the principles of parallel processing using shared memory system to reduce the execution time gradually by increasing number of cores. Two investigations were addressed: Matrix Multiplication which represents balanced load-division and Sorting Algorithms as unbalanced load-division with Order up to 20000. The obtained results are related with analyzing the average and maximum values of (real consumed CPU and thread total execution) times and CPU usage. These algorithms were implemented using Quasar toolkit .

Rashid Zryan N. , [18] 2012, addressed distributed memory system depending on client/servers principles. His work addressed an improved approach for problem subdivision and design flexible algorithms to communicate efficiently between client-side and servers-side. From the main depended trends are: Distributed Parallel Processing technique depending on message passing interface. The algorithms related with these two case-studies were implemented using Java language.

1.4 Layout of the thesis

This thesis is organized into five chapters as follows:

Chapter 2: Describes the PP techniques in general, with the related techniques of PProg, Parallel Computing (PrC), Shared and distributed memories, client/server, and clusters.

Chapter 3: Deals with the proposed algorithms for improved approach of PP and its Applications.

Chapter 4: Introduces the output results and the related discussions, which are representing the application of the algorithms discussed in chapter three using two major case-studies.

Chapter 5: Illustrates the conclusions of this work, and the future work suggestions.

CHAPTER 2

BACKGROUND THEORY

2.1 Introduction

The PP interested by both architectural and algorithmic methods in order to enhance their performance or other elements, such as cost-effectiveness, and reliability of digital computers through different forms of concurrency. Although, concurrent computation has been existed till the earlier of digital computers, it has been applied recently in a manner, and on a scale that leads to a better performance, or a higher cost-effectiveness, comparing with vector supercomputers. Furthermore, motivation is required to the study of parallel architectures and algorithms [3].

2.2 Complex problems and their processing

The PP deals with issues in a centralized manner. This means that the information of process requires transition to such location so that the parallel system is existed. On the contrary, process, which is distributed, deals with the information in a distributed manner [19].

The techniques, such as processing of parallel and distribution, have developed as a relevant field of computation. Moreover, the solution of large-scale problems with the technique that previously explored may be outilined their main process as follows: Firstly, the problems for both parallel and distributed processing formulations. Secondly, the development of specific parallel and distributed algorithms may solve the large scale problems. Thirdly, the algorithm complexity and performance are evluated in order to measure the precise and the reliability of the suggested solution [19].

2.3 Parallel processing

The term PP is the process of taking the responsibility of a big task without utilizing the computer feeding. Since the big task may take long time to accomplish, it is preferred to disperse into small subtasks in order to go on simultaneously .Ultimately, The larg-task accomplishment in short time with dividing into smaller tasks is the aim of division-and-conquer approach instead of processing the whole large task altogether. Furthermore, multiple processors and disks are used in PS in order to improve the process and I/O speeds. The latter assist multiple processors to be capable of working on various parts of task at the same time in order to accomplish thereof in faster way than could be done in vise versa. In addition, database processing works well with parallelism [20].

2.3.1 Taxonomy of computer architecture

Flynn, 1966, classification scheme, who defined the main popular taxonomy of computer architecture, is related to the concept of a stream information. Moreover, the information of instruction and data flow into a processor. Firstly, the instruction stream is the instruction sequences that is fulfilled by the unit processing. Secondly, the data stream is also known as traffic data is the alteration between the memory and the unit of processing. Consequently, the instruction or data streams can be single or multiple depending on Flynn [5].

Conventional single-processor von Neumann computers are classified as Single-Instruction Single-Data (SISD) systems. In addition, PrC is classified in both as a Single-Instruction, or Multiple-Data (SIMD) or Multiple-Instruction. Multiple - Data (MIMD) is defined that there is solely one control unit and the same instruction executed all processors in a synchronized fashion [5].

The PrC with using and proposing many various architectural alternatives; have been utilized for a long period of time. Generally, a parallel computer, which is described as a collection of elements processing, is capable of solving large problems quickly with the characteristic of communication and cooperation. In addition, this latter definition is deliberately quite vague in order to seizure the main different of parallel platforms. Nonetheless, the definition has not addressed some essential detailsm including, the interconnection network's structure and the work's coordination between the processing elements, and essential characteristics of the problems so that to be solved [7].PrC characterize based on the global control, the resultant of data, and control flows. Four categories are recognized [7]:

1. Single-Instruction Single-Data (SISD): a serial (non-parallel) computer and CPU acts on single instruction stream per cycle, only one-data item is being used at input each cycle [21] . The processing element in every step loads an instruction, the data correspondence, and the instruction execution. Consequently, the results are stored in the data storage. Hence, SISD is the convention of the serial computer according to the von Neumann model [7].

Table 2.1: Examples of (SISD)

Load X
Load Y
Z=X+Y
Store Z
X=Y*2
Store X

Time

2. Multiple-Instruction, Single-Data (MISD):

   MISD consists of multiple processing elements, and those elements has their own private program memory, although in each step MISD has solely one common access to a single global data memory [7].

3. SIMD: There are multiple processing elements, that contain a share or distribute access to a data memory individually. The address of shared and distributed debate have explored in the spaces. However, solely one program memory has its control processors, fetches and dispatches instructions [7].

Table 2.2: Examples of (SIMD)

Load X(1)	Load X(2)	Load X(3)
Load X(1)	Load Y(2)	Load Y(3)
Z(1)=X(1)+Y(1)	Z(2)=X(2)+Y(2)	Z(3)=X(3)+Y(3)
Store Z(1)	Store Z(2)	Store Z(3)
X(1)=2*Y(1)	X(2)=2*Y(2)	X(3)=2*Y(3)
Store X(1)	Store X(2)	Store X(3)

4. MIMD: Each element of processing, which is multiple, is separated in instruction and data access so that to (share or disperse) program and data memory. Moreover, every step of processing element loads an instruction and data separately, which the instruction of the data element applies. The example for the MIMD model is Multi core processors or cluster systems. The advantage of SIMD computers with comparison to MIMD computers are easily programmed [7].

Table 2.3: Examples of (MIMD)

P1	P2	P3
Load X(1)	X=Y*Z	Z=X+Y
Load Y(1)	Sumx=Sumx+X	K=min(Z,Y)
Z(1)=X(1)+Y(1)	If (Sumx>0.0)	K=myfun(Y)
Store Z(1)	Call sub Z(2)	K=K*K

2.3.2 Related types of computer architecture techniques

Computer architecture have always endeavor to raise their computer performance. Their fulfilment may come from Furthermore, the fast dense circuitry, packaging technology, and parallelism may be come from the computer fulfilment [5]**.**

a. Symmetric multiprocessor

SMP system is a process that execute threateness comprising kernel code, application code, and interrupt service code. The fulfilment of the SMP system, which permit any processor used entirely and individually, is been done through with exactly the same processors and interconnection, combined with appropriate application and OS kernel design. Consequently, the high processing density, low cost per MIPS, and increase scalability are the main advantages of SMP [22].

b. Single Program Multiple Data (SPMD)

SPMD is the execution of entire processors in parallogram by solely one parallel program. Furthermore, the execution of program is performed concurrently with participating the processors. The parallelism data is the resultant of utilizing

SPMD model, and this may occure when each processor taken part in the structural data [7]. Furthermore, A single program multiple data programming style is that a number of different processes or threads fulfill the same computation on various sets of data [23].

2.3.3 Pipelining

It is an implementation technique where overlapped by multiple instructions in the execution. The parallelism advantages are been taken with existing over the action requirement so that to execute an instruction. Nowadays, pipelining is considedered the main techniques od implementation of using to raise the CPUs speed. The shcedule of pipelining hardware is the concept behind the inheritance of pipelining software[24]. Pipelining considers a very efficient technique for improving system throughput, which is the rate of completion task per unit time. Hence, two conditions are required for raising the influence of the technique [2].

The aim of a pipeline system is to decrease the time of delay that the computer processor is affected on the delay waiting in order to complete the instruction Whereas,the current instruction is executing during a pipeline design, the processor starts the execution of the next instruction. For that reason, The overlap execute with the different phases of instruction. This caused in order to protect the pipeline fully with as many execution sequences as possible [25].

The time declinement between clocked circuits with reducing the quality of logic per stage in order to execute pipelining, as well as the practical limitation of stage number with the decomposition of instruction processing [26].

The use of instruction pipelining is identical to an assembly line in term of a manufacturing plant. An assembly line has an advantage of the fact by going product through various stages of production. Moreover, the production process, which is laid out in an assembly line products at different stages, can be operated meaningwhile. The latter process is also referred to pipelining, because the initial inputs are accepted at only one end even before accepting previous inputs appear as outputs at the other end [6].

Pipelining in microprocessors works the same as assembly lines in manufacturing according to the purpose: The details about the last product do not require to know by the workers (functional units), although it is required to be highly skilled and specialized for solely one task. The same chore is executed by the worker respectively on various objects with handing the half finished product to the next worker in line [27].

2.4 Why parallel processing?

It seem that the quest unfinished for higher-performance digital computers. In the past two decades, there was an exponential growth by the performance of microprocessors [3].

The architectural properties of introduction, and improvements like on-chip cache memories, big instruction buffers, multiple instruction issues per cycle, multithreading, deep pipelines, expired instruction execution, and prediction of branch [3].

2.5 Parallel and distributed processing systems

Parallel and distributed processing is considered one of the vast topic that has been researched for several decades. Moreover, conventional supercomputers with having solely one single processor are fast in speed, but however, they are very expensive and their performance relies on their memory bandwidth. This issues have been resolved by time and the technology development, and hence conventional supercomputers have been replaced with computers, which are cheap in price, many parallel power, disperse of process resources [19].

2.5.1 Parallel systems

The physical arrangement is represented by PS for PP and parallel machine and computer network are types of PS. And moreover, number of processors are comprised that connected closely to a small physical space by both type of PS. There are many types of parallel machines available in the market. The use of computer network for parallel computation that with the whole network computers represent and be the processors meaning while a virtual parallel machine [19].

2.5.2 Distributed Systems (DS)

The DS is the arrangement of physics for process diepersion. DS is almost similar to a parallel system, but although DS is various in term of geographical division oer a large areas. It is also not compulsorily the DS computers to be similar and could also be heterogenous. The information acquisition is the main advantage of ; for instance, DS might be a network of sensors, which is used for measuring the environment, where a giographical set of sensors dispersion might get the environmental information state. The systems of large scale controlling and computating like reserving airelines and anticipating weathers [19].

In DS , inter-process communication is generally subject to various delays, so, state messages can arrive to synchronizers in unpredictable order[28].

2.6 Parallel programming

The PProg is considered as an essential aspect of high-fulfilement computing scientific and working a niche within the whole products of hardware and software field is the advantages of PProg. Although, more recently PProg instead of working to be a niche, a mainstream of software development techniques is taken place due to a radical change in hardware technology [7].

PProg is for applications to enjoy a continued increase in speed in future hardware generations. One might ask why applications will continue to demand increased speed. Many applications that we have today seem to be running quite fast enough [29].

2.6.1 Implementing parallel programming

PProg is the design of a program or of a parallel algorithm intended for an assumed application problematic. Tasks is the design, which can start with the decomposition of the computations of an application into numerous portions, and it is possible to be computed in parallel on the processors of the parallel hardware or the cores. Parallel programming languages plus environments with the aim of simplifying parallel programming by given that sustenance at the correct level of abstraction [7].

The PProg is expressive enough to permit the specification of many parallel algorithms, is easy to use, and leads to efficient programs. Moreover, the more transparent its implementation is, the easier it is likely to be for the programmer to understand how to obtain good performance. Unfortunately, there are trade-offs between these goals and parallel programming APIs differ in the features provided and in the manner and complexity of their implementation [23].

2.7 Clusters

An important and relatively recent development computer system design is clustering. Clustering is an alternative to SMP as an approach to providing high performance and high availability and is particularly attractive for server applications. We can define a cluster as collection of interconnected, all computers operating together as a united computing resource, which can produce the desired impression of being one machine [6].

Clustering shows four benefits and can moreover be supposed of as design or objectives requirements [6]:

• Absolute scalability: there is the option to make large clusters that far exceed the power of the largest standalone machines.

• Incremental scalability: there is the possibility to add a new desired systems to the cluster however in small increments .

• High availability: Each node in a cluster is described as a standalone computer. Any failure or problem of one node does not lead to the damage or loss of service. There are a lot of products, which the fault tolerance is touched automatically in software.

• Superior price/performance: commodity building blocks make it probable to put together a cluster with equal or bigger computing power than a single large machine, moreover at greatly lower price.

Clusters are very popular in the high performance computing community. A cluster consists of several cheap computers (nodes) linked together. The simplest case is the combination of several desktop computers - known as a network of workstations (NOW). Most of the time, SMP systems (usually dual-CPU system with Intel or AMD CPUs) are used because of their good value for money. They form hybrid systems. The nodes, which are themselves shared memory systems, form a distributed memory system [30].

2.8 Client/Server principles

A Client/Server is a DS, which the application is distributed into at least two parts:
It is possible that one or more servers perform the desired one part and the other wanted part is performed by one or more clients. The clients are linked directly to the servers by some kind of network. It is possible that a client computer could do more than only display data retrieved from the server. Using the desired data, this is provided by the server a sophisticated client possibly run a full application. As two-tier or three-tier a Client/Server systems are regularly classified. A two-tier system splits clients from servers .The whole clients stand on one tier, in addition all servers stand on the second tier [5].

Client/server model of computing in its architecture as does many distributed systems (with the noted exception of peer-to-peer [31].Clients and servers can use hardware and software uniquely suited to the required functions. In particular, front-end and back-end systems normally require computing resources that differ in type and power. Database management systems can employ hardware specifically designed for queries, while graphics functions can employ memory and computing resources that can generate and display intricate diagrams [9].

2.9 Parallel computing

Software usually has been designed for serial computation [21]:

- A single CPU should be able to run on a single computer.
- Each problem should be broken into a separated series of instructions.
- Typically every instruction will be performed one after another.

- There will be only one instruction, which may effect at any moment in time.
- Solving a computational problem there is the need to use Parallel Computing, which is the simultaneous use of multiple compute resources [21]:To be run using multiple processors.
- A problem is broken into discrete parts that can be solved concurrently.
- Each part is further broken down to a series of instructions.
- Instructions from each part execute simultaneously on different processors.
- An overall control/coordination mechanism is employed.

2.9.1 Approaches to parallel computing

In general, there are three main approaches of parallel processing systems according to the organization of their memories:

1. Shared Memory Systems (SMS)

The major category of multiprocessors will be formed by the SMS. Accordingly in the category, the whole processors share a global memory. The way of communication between tasks is based on diverse processors, which is prepared via writing to and reading from the global memory. shown in Figure 2.1. [5].

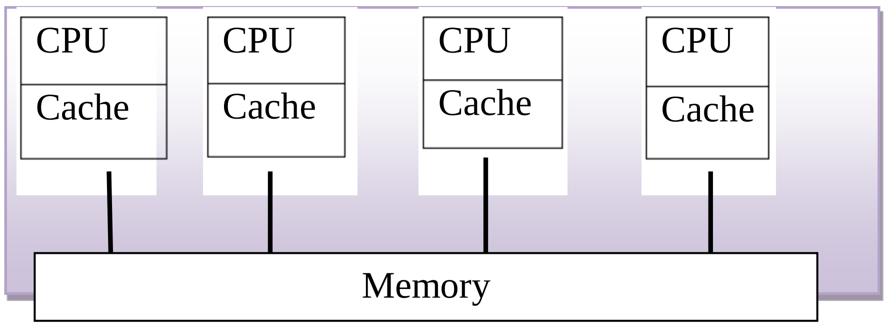


Figure 2.1. Shared Memory Systems

2. Distributed Memory Systems(DMS)

Like SMS , DMS share a common characteristic and at the same time vary widely. DMS need a typical communication network to be able to connect inter-processor memory. However processors have got their own local memory. Across all processors there is no concept of global address space because the memory addresses in one processor do not map to another processor. Each processor works independently since each of them has its own local memory. Any changes to its local memory have no influence on the memory of other processors. Following the concept of cache coherency does not apply to it .The programmer decides how and when data is communicated, if a processor needs access to data in another processor. It is the programmer's responsibility to manage between tasks [21].

Distributed memory is an important advantage of PS over uniprocessor workstations, since it enables them to run much larger problems and to run them much faster. However, the super linear speedup does not allow us to separate out the two effects, and as such does not help us evaluate the effectiveness of the machine’s communication architecture. Finally, using the same problem size (input configuration) as the number of processors is changed often does not reflect realistic usage of the machine[32].

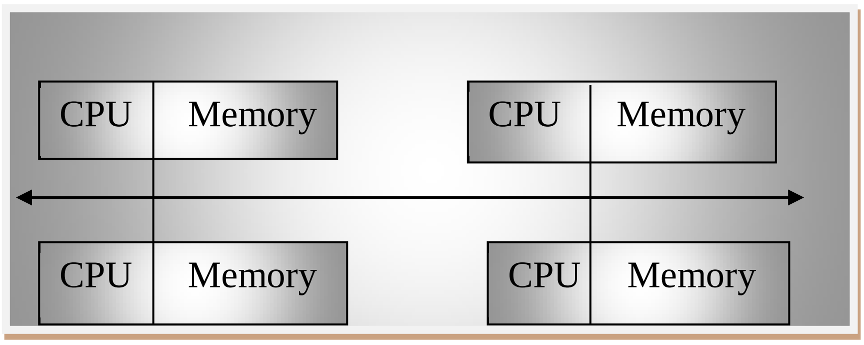


Figure 2.2. Distributed Memory Systems

3. Hybrid distributed-shared memory

It is considered the fastest and biggest in size computers all over the world nowadays which is woked by both shaared and distributed architectural memory. The component shared memory is also a shared memory of machine and/or Graphics Processing Units (GPU). Moreover, the component of distributed memory is the machines of shared memory or GPU of multiple networking, as well as the memory is familiar with its own memory. Hence, shifting data from a machine to another can not be done with the use of network communications [21].

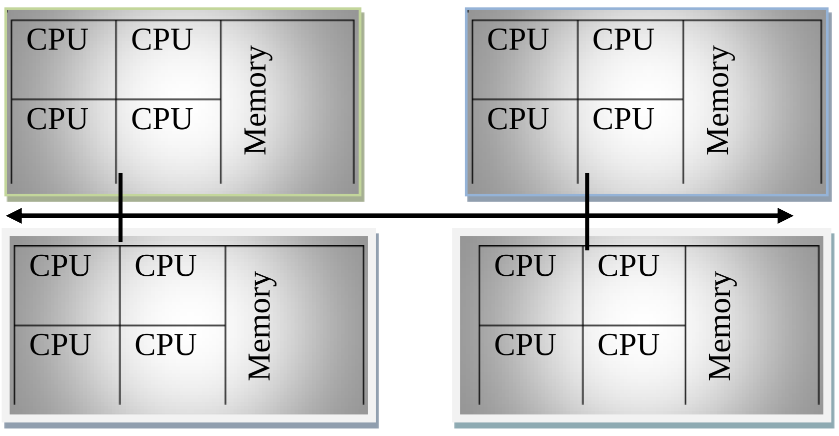


Figure 2.3. Hybrid Distributed-Shared Memory

2.9.2 Types of parallel computing

The connection of parallel computing are been done in such a way that the information exchanged between processes or threads. The latter connection relies on the memory organization of the hardware, parallel computing kinds [7].

a. Parallel computing in software

In order to supply static and dynanmic balance of loading with application with the existance of several packages of software. The different implementation with high-quality of algorithm partision is accessed by application developers with using software packages. Those packages with functional support with the use of balancing load are mostly required by applications. The acknowledge and implementation of partitioning algorithm attemption of developer applications are saved with the use of Packages, however, the comparison between partitioning strategies and their applications are permitted [33].

b. Parallel computing in hardware

The PProg is considered as an essential aspect of high-fulfilement computing scientific and working a niche within the whole products of hardware and software field is the advantages of PProg. Although, more recently PProg instead of working to be a niche, a mainstream of software development techniques is taken place due to a radical change in hardware technology [7].

2.9.3 Classes of parallel computing

In order get a good idea about parallel computing, this section will deal with five famous parallel computing classes.

a. Multi-core computing

It is a system with more than one core on a single chip, and this system is utilized to program the MPMD (multiple program multiple data) in non uniform multi-core system. Various program are run and various set of data are worked in the MPMD model [34].

b. Symmetric multiprocessing

Inside the communication space, two variants of SMP have illustrated. The first variant contains multi core processors, which execute coupled SMP on a single chip firmly. However, the second variant, in order to implement concurrently on a single processor core, multiple tasks or threadsare permitted by simultaneous multithreading[22]. Usually all processors are identical and have equal memory access. This is called symmetric multiprocessing [30].

c. Distributed computing

Distributed computing is a process that connect a set of computers with the use of network, which is used in order to figure out an individual large problem. With more organizing the system the speed interconnection of local area network is increased and consequently the power of a single high-fulfilment computer may be exceeded by many general workstation with the combination of computational resources. Distributed computing can facilitate collaborative work. To be elective distributed computing requires high communication speeds [35].

d. Cluster computing

Cluster Computing consists of more than two computers and used to implement problem or section that has given. Usually, the computers are tied together in a computer cluster by the interconnection network, which is known as a local Area Network (LAN). The computer communaction in the cluster are being done among each other and a shared memory. For that reason, packets over the LAN is mainly used to communicate the processors in a cluster [2].

e. Grid computing

Grid computing refers to provide an accessability of computing resources, which is distributed over a Wide Area Network (WAN). Hence, distributing a collection of huge quantity of processors over a broaden a geographic area. A grid computer is capable of handling large - scale of computational issues; for instance, N - body simulations, seismic simulations, and atmospheric and oceanic simulations. In comparison with computer cluster, a large cluster with exchanging the LAN to WAN; for example the internet [2].

2.10 Massage-passing systems

Message Passing Systems are multiprocessor class systems that any of this process has an accessability to go through its local memory. Opposite to shared memory systems, The fulfilment of message passing communication systems by sned and receive operations [5].

Massage-Passing, which is a collection of tasks, utilize its own local memory during computation. The exchange of data are occurred through sending and receiving messages. Moreover, the cooperative operations are required in order to transfer data, which is fulfilled by every process. For instance, a send and receive operations must be matched [21].

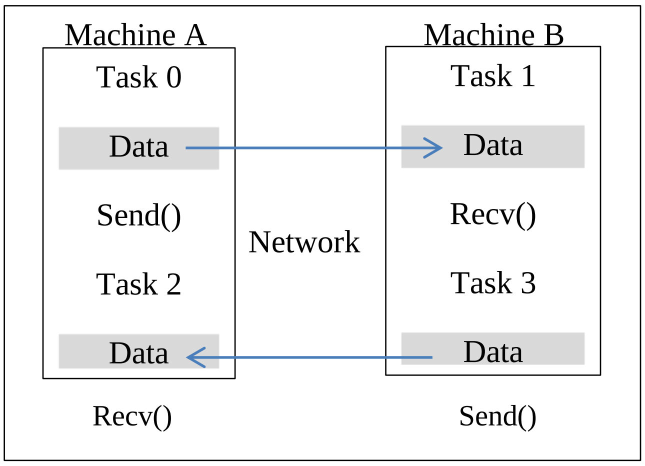


Figure 2.4. Massage-Passing

CHAPTER 3

STRUCTURE AND ALGORITHMS OF THE PROPOSED SYSTEM

3.1 General view

The SM is a common inter-processor communication paradigm for a single-chip multi-processor platforms. All processors are connected to a common memory RAM. Moreover, the big advantage of SM systems is that all the processors can make useful for the entire memory. As a result, this helps the simplicity of programming and useful in efficiency. Furthermore, the factor limitations of their performance are the number of processors and memory modules that can be connected to each other. Due to these, SM-systems usually consist of rather few processors.

On the other hand, in a distributed memory system, each process, which has its own address space, communicates with other processes by MP as (sending and receiving messages). In addition, a processor, which is connected to each other, has its own local memory. Furthermore, there is no limitation on number of processors and memory modules in a distributed memory system, because of servers, which are connected as cluster-network, can be extended to any required number. Consequently, the second technique in this work depends on the number of servers, which are 8-servers of identical properties in the cluster-network.

The proposed algorithms have two main parts; first one related to hardware of the work, and the second is about the software that guides the components of hardware and manages the passing of messages between Client and Servers with deferent cases.

3.2 Hardware part

The hardware is consisted of client-side, and servers-side. In addition, the network that contains from both sides is designed according to star topology. In such works, the properties of computers, which are different or similar from one computer to another, are essential. This means having identical-computers. In fact, for more accurate results with acceptable comparisons, it is preferred to depend on identical-computers. Therefore, in this work all computers in both sides, which have the following properties: (CPU: Core 2 Due, Speed 2.4 GHz, RAM: 4 GByte, and HD:500 GByte), are identical completely.

Furthermore, client-side has only one host, which controls the sending of MP as operations to other side. Client-host contains the main program that can treats with all server-hosts in individual, subgroups, or all of them. The secondary storage of the client-host contains the original data on each case of study that sends to the other side, and then receives the results, which were calculated by the servers-side.

Servers-side consists of eight hosts (or sockets) connected in such a way in order to get a cluster of 8-sockets. Moreover, each socket contains a program that has an ability of receiving data, making the requirement of processing, calculating the results, and then sending them to the client-side. Furthermore, servers-side can also store the received data and determined the results on secondary storages, or send them directly to the client-side.

3.3 Software part

The software part same as hardware part also consists of two sides that are client-side-software and servers-side-software.

Client-side-software , represents the main-program, which is responsible of the following tasks:

1. Detecting the number of connected server-sockets on other side.
2. Deciding the number of server-hosts that will receive the messages from the client.
3. Sending control-messages to server-sockets.

4. Sending related data (as message-text or as data-files) to server-socket.
5. Monitoring all related server-sockets in case if they send any result or any query-message.
6. Responding the query-messages that received from other side.
7. Receiving the calculated results by server-sockets and accumulating in order to get the final results.
8. Making sure that all sending or receiving messages and data are stored on the Client-side in secondary storages.

On the other hand, servers-side-software represents the programs that serve the commands issued from the main program (i.e. client program). Moreover, the software at each server-host is responsible of the following tasks:

1. Detecting the connection status of the client-host.
2. Deciding to work as which socket, according to the number of server-sockets, sent by the client, taking into account that it may be out of work for certain numbers of server-socket. For example; if the number of server-sockets is 2, then only servers (1 & 2) will work.
3. Receiving the control-messages from client-host and guiding the execution of the server-program so that to apply the client-requirements.
4. Receiving the related data (as message-text or as data-files) from client-socket.
5. Monitoring client-host in such case if it sends any immediate command, message, or data.
6. Run the appropriate-subroutines according to the requirements of client-host and calculate the correct results, knowing that each server will treat with part of data, which is selected for server by the client.
7. Sending the calculated results to client-host, knowing that these results will be arranged in a form in order to be managed by the client-host in a suitable manner.
8. Each server-program contains all subroutines of the same case study. This gives the server-program the ability of treating with any selected part of data and chose the appropriate subroutine to calculate the required results.

3.4 Messages transferred between client-side and servers-side

There are two types of messages, which are (control-messages and data-messages), related to the PP approaches.

3.4.1 Control messages

Control messages, which is issued by client-host, is sent to servers-sockets. These messages control the management of the processing of the entire network and monitor the performance of the hosts, especially servers-hosts.

The structure of control-messages can be implemented through the following steps:

1. Connection status of each server-socket, either it is ready or not
2. Send a message for each unready server to open its embedded connection link
3. Selecting number of server-sockets to be participate with task
4. Send a control message to connect the selected servers to be able for receiving data receiving data
5. Acknowledgment from the selected server
6. Sending the starting-signal and/or termination-signal for any selected server-socket
7. Return termination signals from the selected servers
8. Tear-down signal from client to terminate the connection

3.4.2 Data messages

Data messages , issued by client-side and/or servers-side. These messages carry specific data, which help running processes at server-sockets if the messages are issued by client-host. In addition, data messages may represent specific results, when the messages are issued by the server-sockets.

The structure of data-messages can be implemented through the following steps:

1. Connection status occurs between client-side and servers-side
2. Starting task-time (issued by client)
3. Starting CPU-time (issued by client)
4. Size of data-arrays that must be generate by client and used by servers
5. Names of files containing these data-arrays stored in a shared-drive to be use by both client-side and servers-side
6. Starting running time (issued by each server)
7. Starting CPU time (issued by each server)
8. Idling CPU time (issued by each server)
9. Size of data-arrays that must be assigning by servers after processing and used by client later for rearrangement
10. Names of files created by client and containing these data-arrays to be use by both client-side and servers-side
11. Terminating CPU-time (issued by each server)
12. Terminating running-time (issued by each server)
13. Consumed CPU-time (issued by each server)
14. Consumed running time (calculated by each server)
15. CPU usage percentage ratio (calculated by each server), then:
    - CPU idling percentage ratio (calculated by each server)
    - Terminating CPU-time (issued by client)
    - Terminating task-time (issued by client)
    - Consumed CPU-time (calculated by client)
    - Consumed task-time (calculated by client)

3.5 Matrix algebra case study

In order to observe the importance of PP techniques, researchers usually depend on one or more cases of study, which explain these algorithms and produce the results with less time of execution comparing with single processing technique.

The latter works are avoided with using the large number of computers as servers (even for testing purposes) because of the following reasons:

1. The complexity of providing the large number of computers in order to operate as servers.
2. Problems of interface-network components of connections.
3. Interfaces the problems between client-side and servers-side.
4. Synchronize the operation of servers with respect to the client (this is a major problem).
5. The difficulty of the program for both client-side-software and servers-side-software.
6. The difficulty of jobs, which are distributed by the client-side among servers-sockets.
7. The difficulty of message-passing operations between both client-side and servers-side.
8. The difficulty of subdividing the problem in the client-side, so that to be solved by a cluster of servers.
9. The difficulty of preparing the servers in order to be able of treating the sub-problems, which are provided by the client.

Matrix algebra is an essential kind of case study related to PP, and the previous works were solely depended on eight servers as maximum. However, eight servers are used in this work, and it can be N-servers according to the capacity of the laboratory of these experiments, the algorithm is designed to treat with two original matrices of square order (4096, 4096). This means that each matrix will contain (16,777,216) elements. Therefore, there will be (33,554,432) elements divided into sub parts (sub-matrices) to perform the (ADDING, SUBTRACTING, and MULTIPLICATION) operations as part of matrix algebra according to the following situations:

1. Apply all operations on one server-socket.
2. Apply all operations on two server-sockets.
3. Apply all operations on four server-sockets.
4. Apply all operations on eight server-sockets.

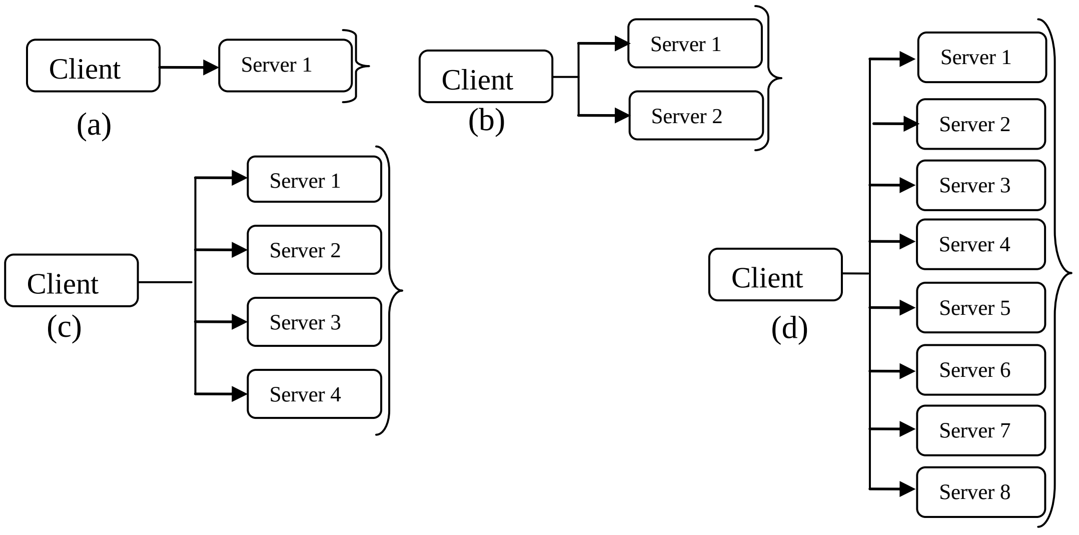


Figure 3.1. Cases of Matrix Algebra Algorithm.
(a) one-server. (b) two-servers. (c) four-servers. (d) eight-servers.

Number of servers=2^m, where m= {0, 1, 2, 3 and 4}

<u>First Case</u>: Addition and Subtraction of two matrices A and B without dividing

$$A \begin{bmatrix} \text{server 1} \end{bmatrix} \pm B \begin{bmatrix} \text{server 1} \end{bmatrix} = C \begin{bmatrix} \text{server 1} \end{bmatrix}$$

<u>Second case</u>: Addition and Subtraction of two matrices A and B after dividing each matrix into two parts

$$A \begin{bmatrix} \underset{A1}{\text{server1}} & \vline & \underset{A2}{\text{server2}} \end{bmatrix} \pm B \begin{bmatrix} \underset{B1}{\text{server1}} & \vline & \underset{B2}{\text{server2}} \end{bmatrix} = C \begin{bmatrix} \underset{C1}{\text{server1}} & \vline & \underset{C2}{\text{server2}} \end{bmatrix}$$

<u>Third case</u>: Addition and Subtraction of two matrices A and B after dividing each matrix into four parts

$$A \left(\begin{array}{c|c} \text{server1} & \text{server2} \\ \hline \text{Server3} & \text{server4} \end{array}\right) \pm B \left(\begin{array}{c|c} \text{server1} & \text{server2} \\ \hline \text{server3} & \text{server4} \end{array}\right) = C \left(\begin{array}{c|c} \text{server1} & \text{server2} \\ \hline \text{server3} & \text{server4} \end{array}\right)$$

Fourth case: Addition and subtraction of two matrices A and B but divided each matrix in to eight parts.

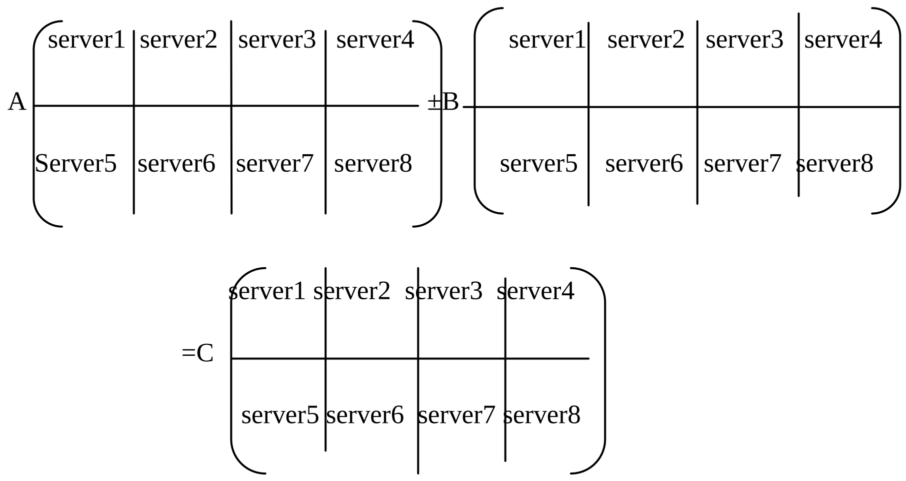
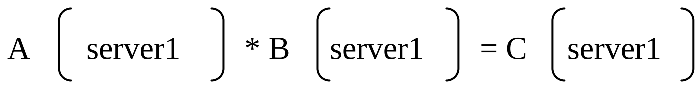
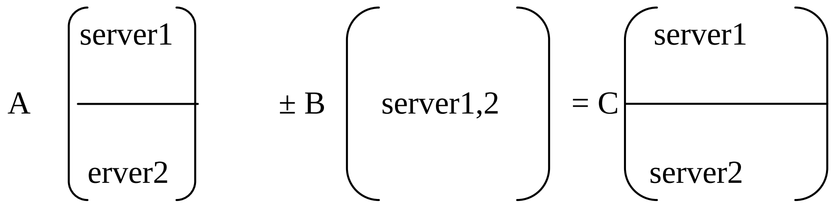


Figure 3.2. All cases of Addition and Subtraction of Matrices

First case: Multiplication of two matrices A and B without divided



Second case: Multiplication of two matrices A and B but divided A matrix in to two parts



Third case: Multiplication of two matrices A and B but divided the matrix A in to four parts.

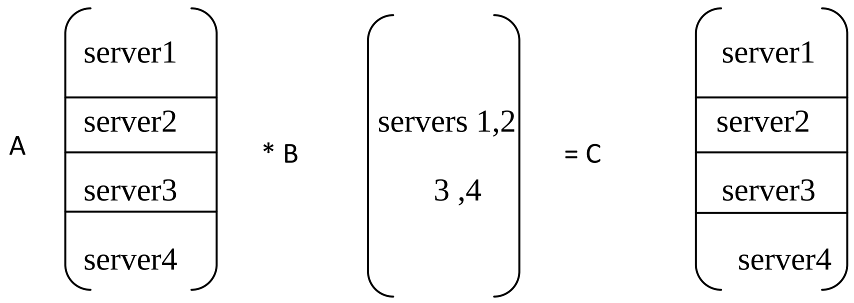


Figure 3.3. Third cases of Multiplication of Matrices (cases 1, 2, 3 ,and 4)

Fourth case: Multiplication of two matrices A and B but divided the matrix A in to eight parts

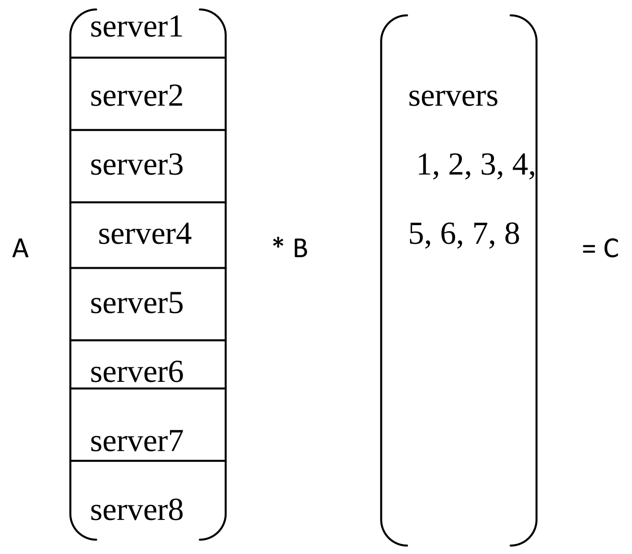


Figure 3.4. Four cases of Multiplication of Matrices (cases 1, 2, 3 ,4 and 8)

3.5.1 Client software part

The structure of client-software-part can be implemented through the following steps:

1. Connection status occurs between client-side and servers-side
1. Input unsorted array elements or read them from certain files at client side
2. Detect all socket-servers connected to the client
3. Record starting running client-times
4. Select No. of servers to do the job.
5. If No=1 then: Send the matrices A and B without dividing them, go to step 9. Else
6. If No=2 Send the matrices A and B with dividing them into 2 parts, , then go to step 9, else
7. If No=4 Send the matrices A and B with dividing them into 4 parts, , then go to step 9, else

8. If No=8 Send the matrices A and B with dividing them into 8 parts, then go to step 9, else (The No. must be one of {1, 2, 4, and 8 }) then go to step 4.
9. Store the contents of the two matrices and there dimensions in shared files
10. Send the files names to servers-side
11. Receive CPU-usage percentage, running time and CPU-idle-percentage
12. Receive the files-names of result matrices from servers-side
13. If No=1 then: There is no need for reassembling the matrices A, B , go to step 17. Else
14. If No=2 .Reassemble the matrices A ,B from 2-resultant-parts
15. If No=4 . Reassemble the matrices A ,B from 4-resultant-parts
16. If No=8 . Reassemble the matrices A ,B from 8-resultant-parts,if (The No. must be one of {1, 2, 4, and 8 }) continue .else if (The No. must be one of {1, 2, 4, and 8 }) go to step 13.
17. Record termination running client-times
18. Consumed running client-times
19. Display assembled matrices
20. CPU-usage percentage, running time and CPU-idle-percentage

3.5.2 Servers software part

The structure of Server-software-part can be implemented through the following steps:

1. Connection status occurs between client-side and servers-side
1. The server-side receives the files names from client-side
2. Load the files from shared-drive and extracts the matrices dimensions and there elements
3. Each server record starting (running, CPU) servers-times
4. Each server perform to calculate the matrices operations such as: (addition, subtraction and multiplication)
5. Each server record terminating (running, CPU) times

6. Each server calculate the consumed (running, CPU) times, (CPU-usage and Idle) percentages
7. Send the results of corresponding matrices and times to separated shared files and send files names to client-side

CHAPTER 4

IMPLEMENTATION RESULTS AND DISCUSSION

4.1 Introduction

In order to illustrate the efficiency of the proposed parallel processing system, the matrix-algebra case-study is depended to be applied on the system that contains one client and eight servers. The results are obtained for four matrix-orders starting with (64*64) and increasing this load yet reaching (4096*4096) as a high-load.

4.2 Matrix algebra case-study

This case-study needs to one client and eight servers for this situation and the algorithms are divided according to the steps illustrated in chapter three. The obtained results are illustrated in the Tables (4.1 to 4.6) and Figures (4.1 to 4.16). Also, the results are divided into two main groups; one of them is related to the average values of average and total execution timings and usages for servers-side which are represents the average of related times or usages for all servers as an acceptable value to be depended, these values are illustrated in Tables (4.1 to 4.4) and plotted as in Figures (4.1 to 4.12).

The second main group of the results is an additional assessment of performance of this work in the view of the latest returning results by the servers-side which is named here as Maximum-Values. These values are shown in Table (4.5 and 4.6) and illustrated in Figures (4.13 to 4.16).

Table 4.1: Average Values For Matrix Order = 64 * 64

	Matrix Order = 64 * 64			
Time (Sec.)	One server	Two servers	Four servers	Eight servers
Elapsed CPU Time of Client	0.2875	0.275	0.325	0.3
Total Execution Time of Client	0.34917	0.347313	0.445618	0.405016
Elapsed Average Consumed CPU Time of Servers	0.0375	0.03125	0.04375	0.035938
Average Total Execution Time of Servers	0.238018	0.189448	0.143308	0.199
Average CPU Usage % of Servers	12%	12.8%	28%	14.4%

Table 4.2 : Average Values of Matrix Algebra-Case For Matrix Order = 256 * 256

	Matrix Order = 256 * 256			
Time (Sec.)	One server	Two servers	Four servers	Eight servers
Elapsed CPU Time of Client	0.85	0.7	0.75	0.8
Total Execution Time of Client	1.012807	0.887018	0.930387	1.010495
Elapsed Average Consumed CPU Time of Servers	0.35	0.275	0.221875	0.215625
Average Total Execution Time of Servers	0.690817	0.533898	0.396176	0.321609
Average CPU Usage % of Servers	40%	40.8%	44%	54.4%

Table 4.3: Average Values of Matrix Algebra-Case For Matrix Order = 1024 * 1024

	Matrix order = 1024 * 1024			
Time (Sec.)	One server	Two servers	Four servers	Eight servers
Elapsed CPU Time of Client	23.7875	15.75	14.7375	15.425
Total Execution Time of Client	25.63199	17.57746	16.58674	17.29236
Elapsed Average Consumed CPU Time of Servers	17.025	10.00625	6.68125	4.948438
Average Total Execution Time of Servers	21.30418	13.04144	8.365478	5.965315
Average CPU Usage % of Servers	63.2%	60.8%	63.2%	65.6%

Table 4.4: Average Values of Matrix Algebra-Case For Matrix Order = 4096 * 4096

	Matrix Order = 4096 * 4096			
Time (Sec.)	One server	Two servers	Four servers	Eight servers
Elapsed CPU Time of Client	1450.713	807.375	569.05	497.55
Total Execution Time of Client	1478.944	836.3778	599.8794	530.8112
Elapsed Average Consumed CPU Time of Servers	1312.912	684.1312	375.1906	222.3969
Average Total Execution Time of Servers	1401.361	758.6677	439.9162	276.198
Average CPU Usage % of Servers	74.4%	72%	71.2%	72%

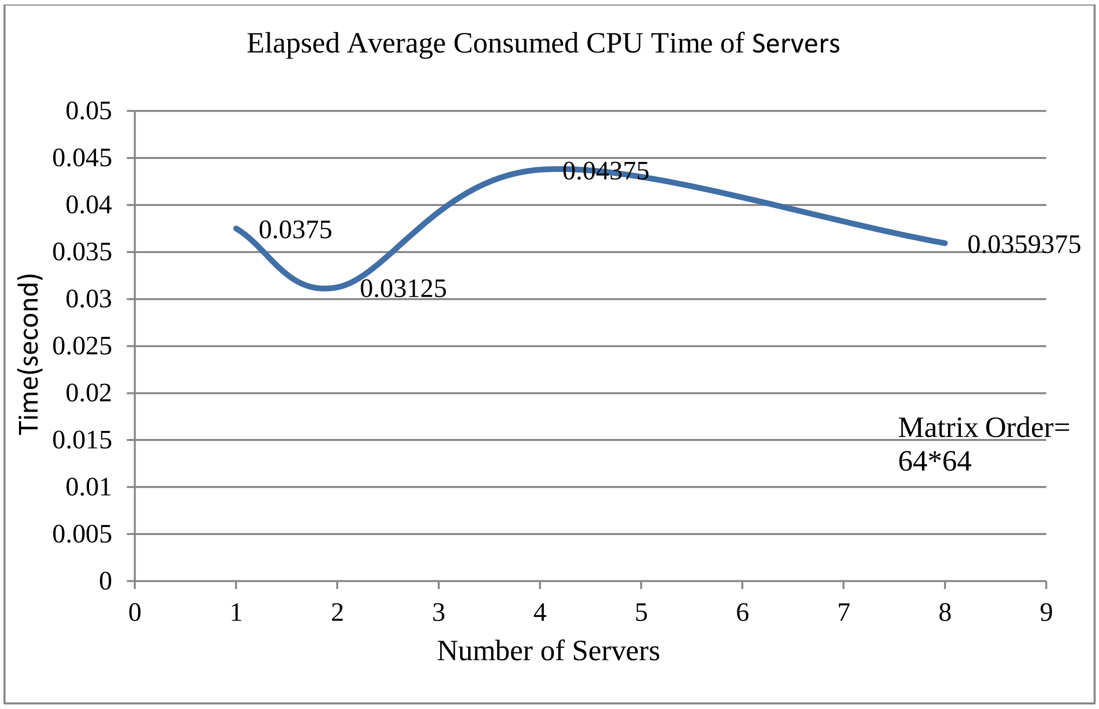


Figure 4.1. Elapsed Average Consumed CPU Time of Servers For Matrix-Algebra of (64*64)

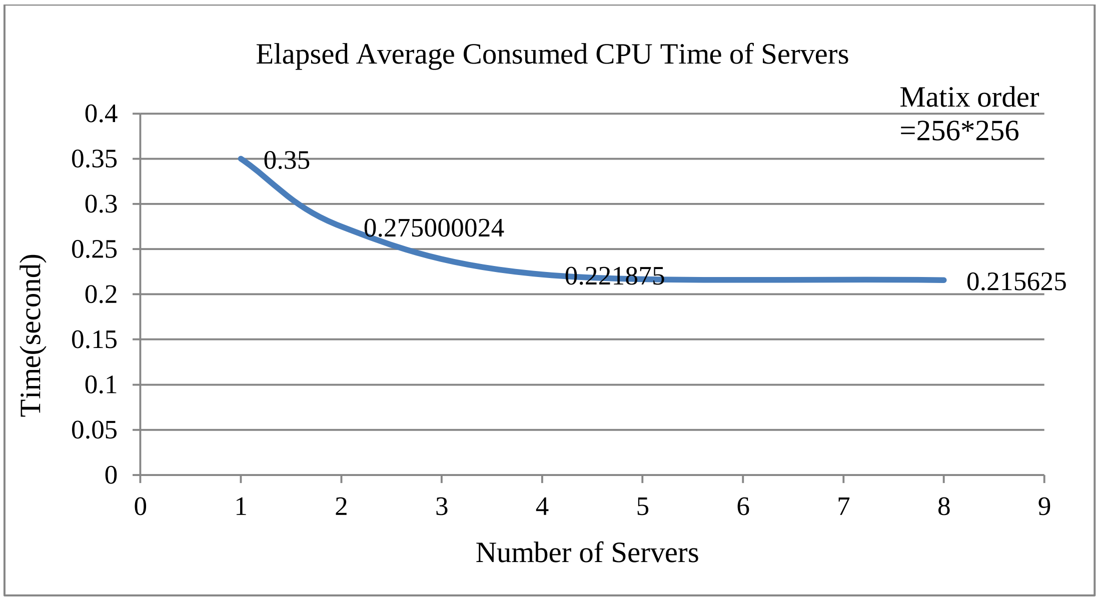


Figure 4.2. Elapsed Average Consumed CPU Time of Servers For Matrix-Algebra of (256*256)

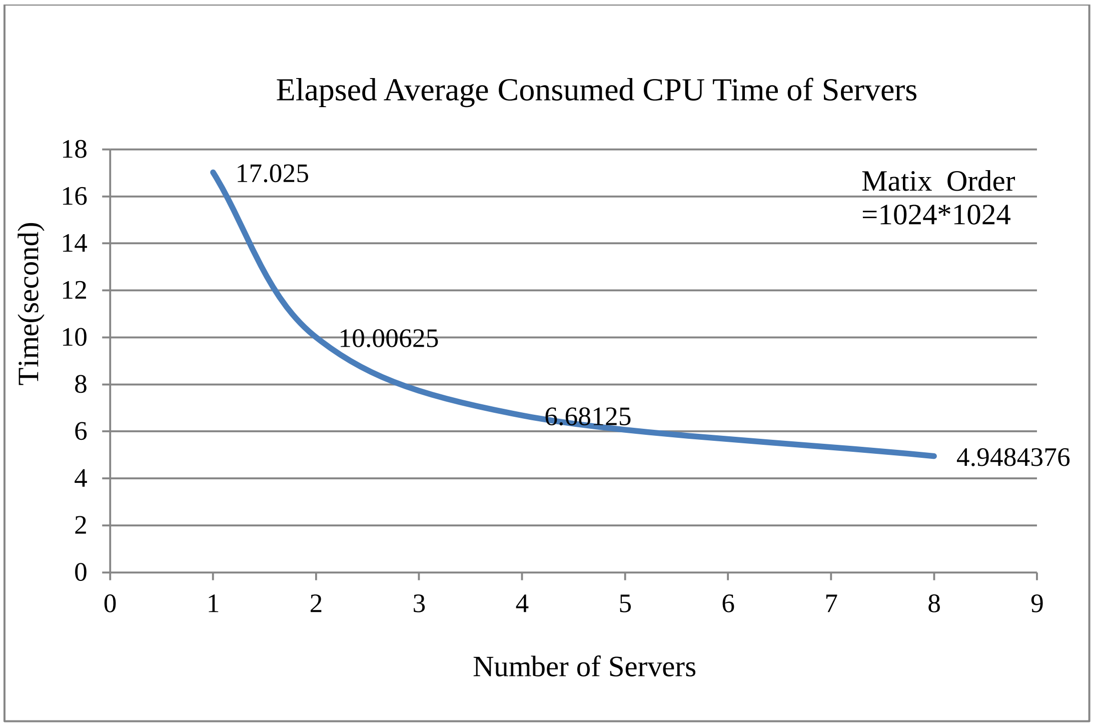


Figure 4.3. Elapsed Average Consumed CPU Time of Servers For Matrix-Algebra of (1024*1024)

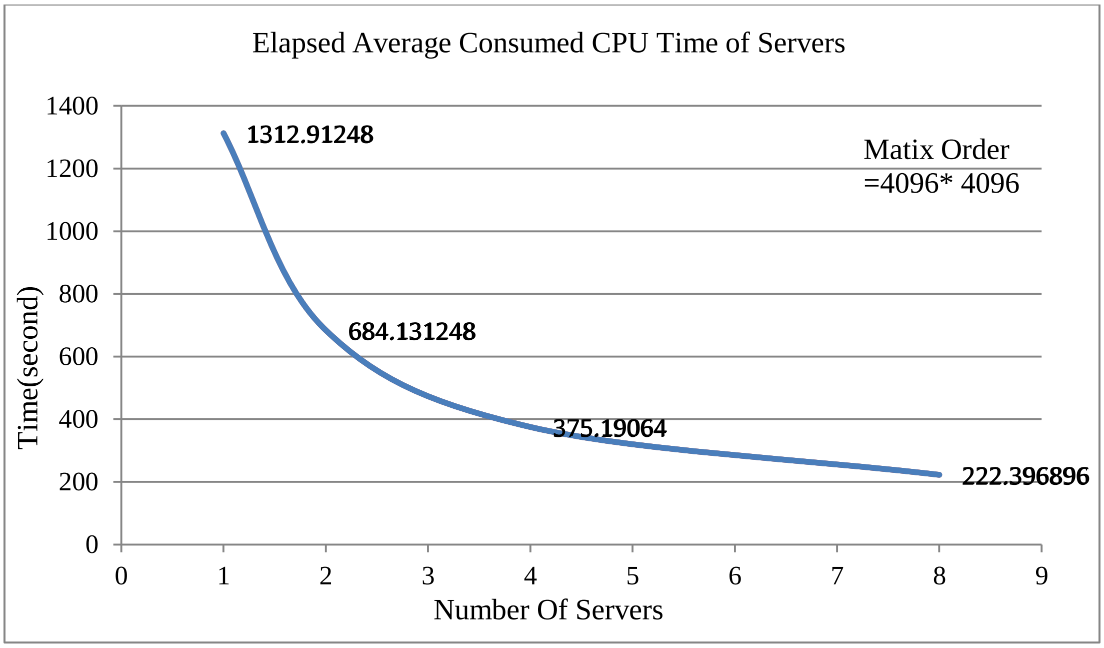


Figure 4.4. Elapsed Average Consumed CPU Time of Servers For Matrix-Algebra of (4096*4096)

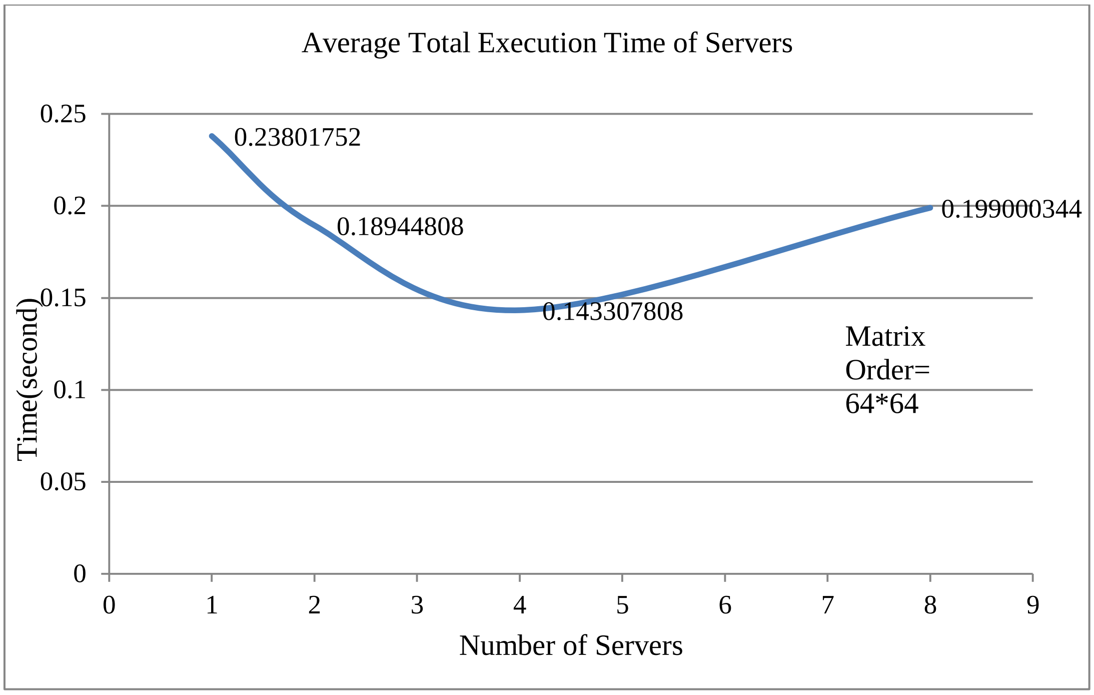


Figure 4.5. Average Total Execution Time of Servers For Matrix-Algebra of (64*64)

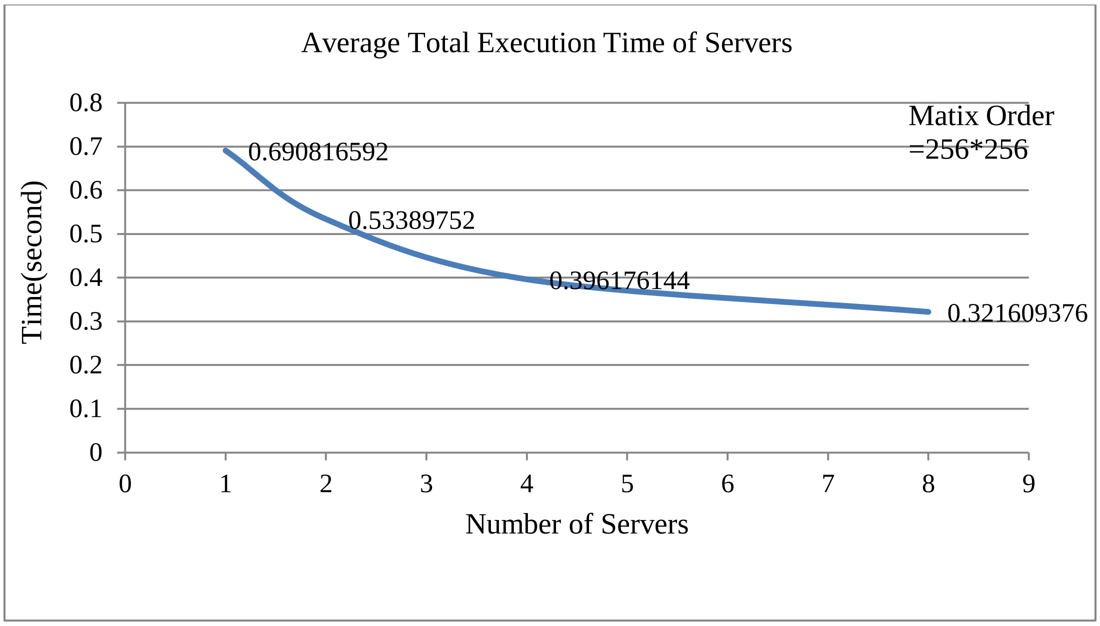


Figure 4.6. Average Total Execution Time of Servers For Matrix-Algebra of (256*256)

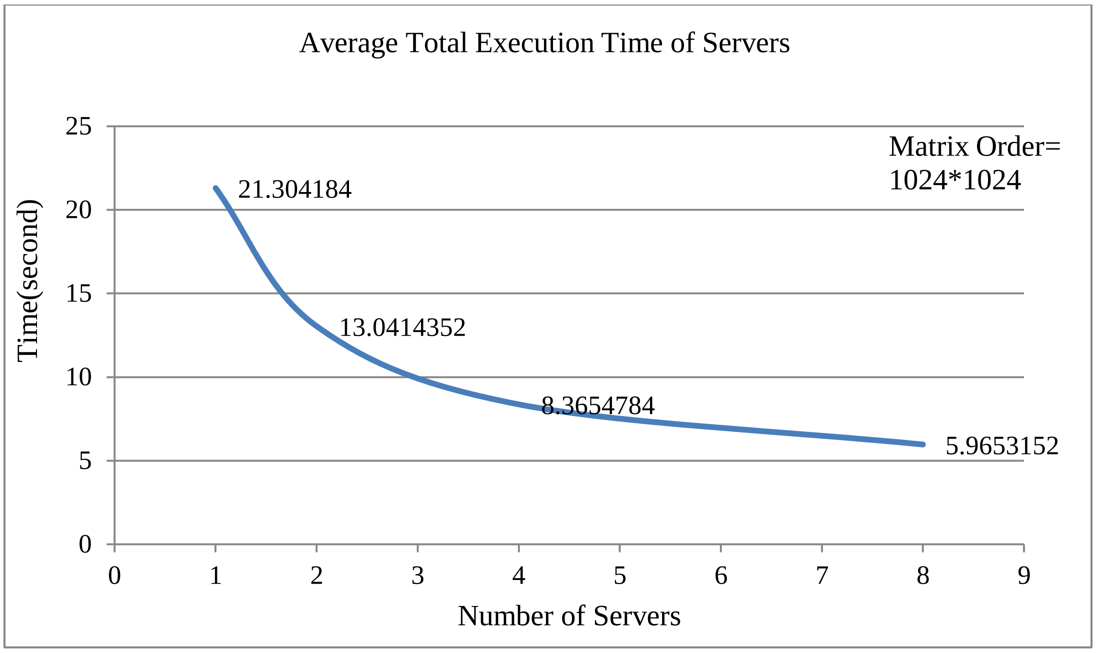


Figure 4.7. Average Total Execution Time of Servers For Matrix-Algebra of (1024*1024)

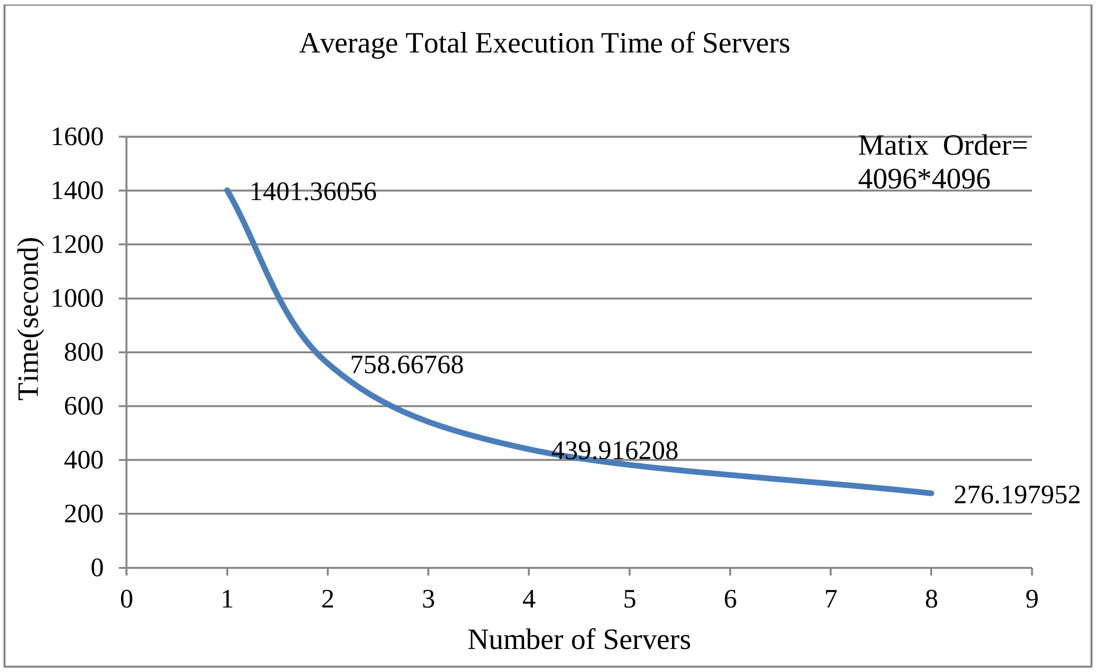


Figure 4.8. Average Total Execution Time of Servers For Matrix-Algebra of (4096*4096)

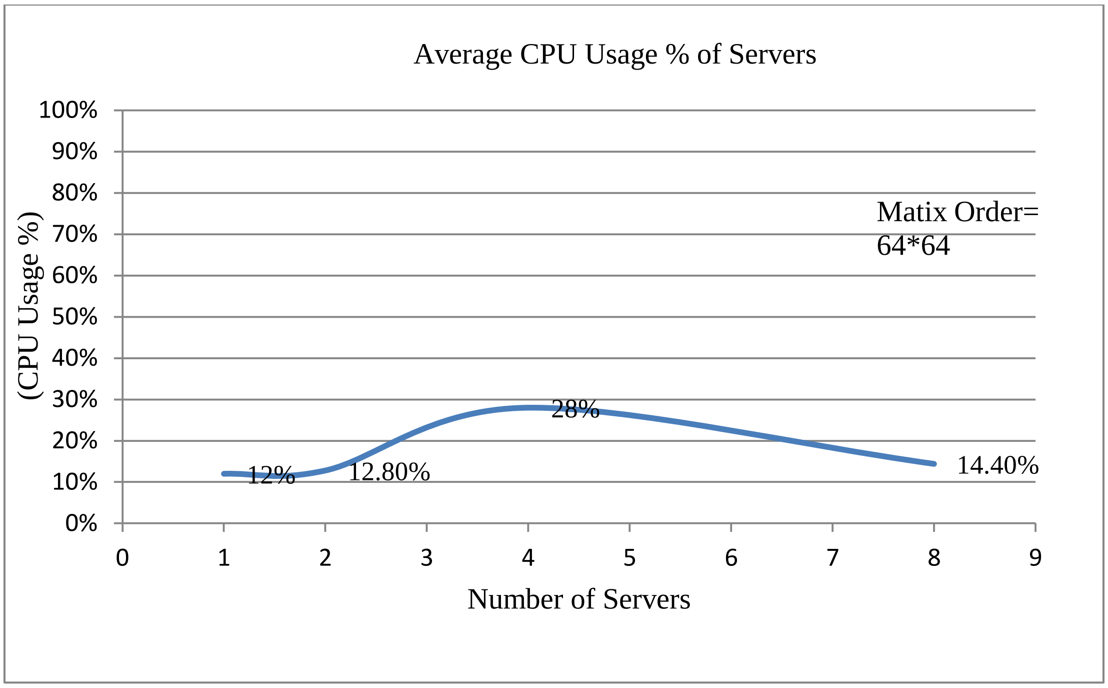


Figure 4.9. Average CPU Usage % of Servers For Matrix-Algebra of(64*64)

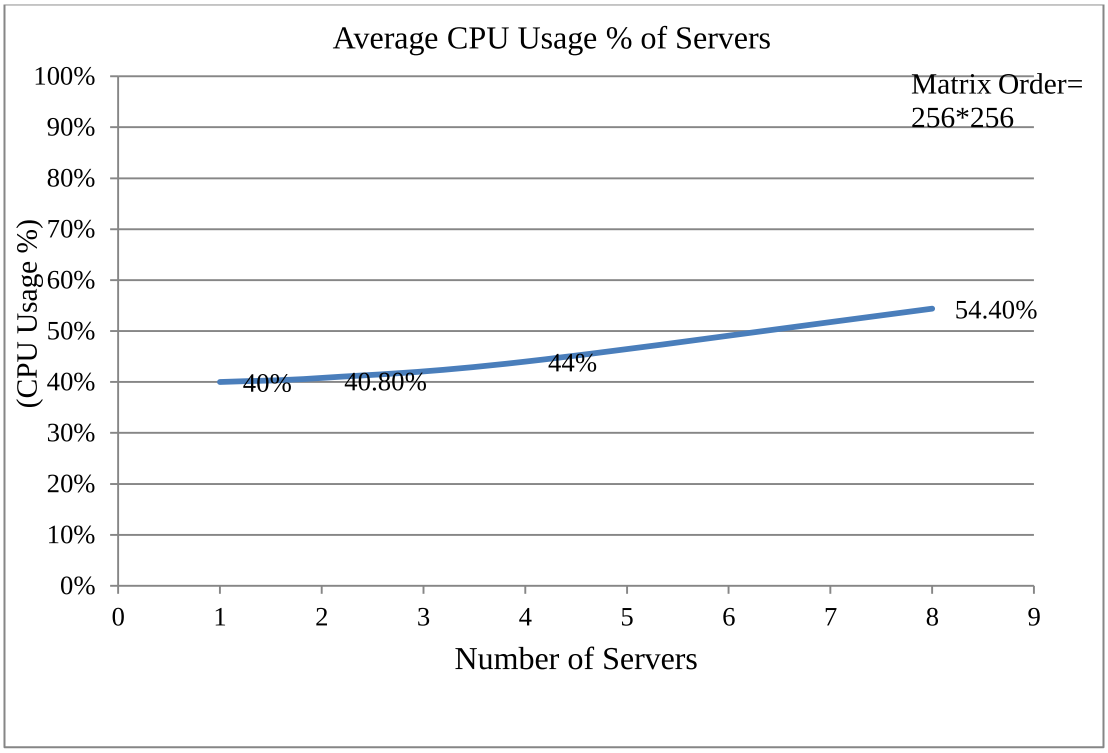


Figure 4.10. Average CPU Usage % of Servers For Matrix-Algebra of(256*256)

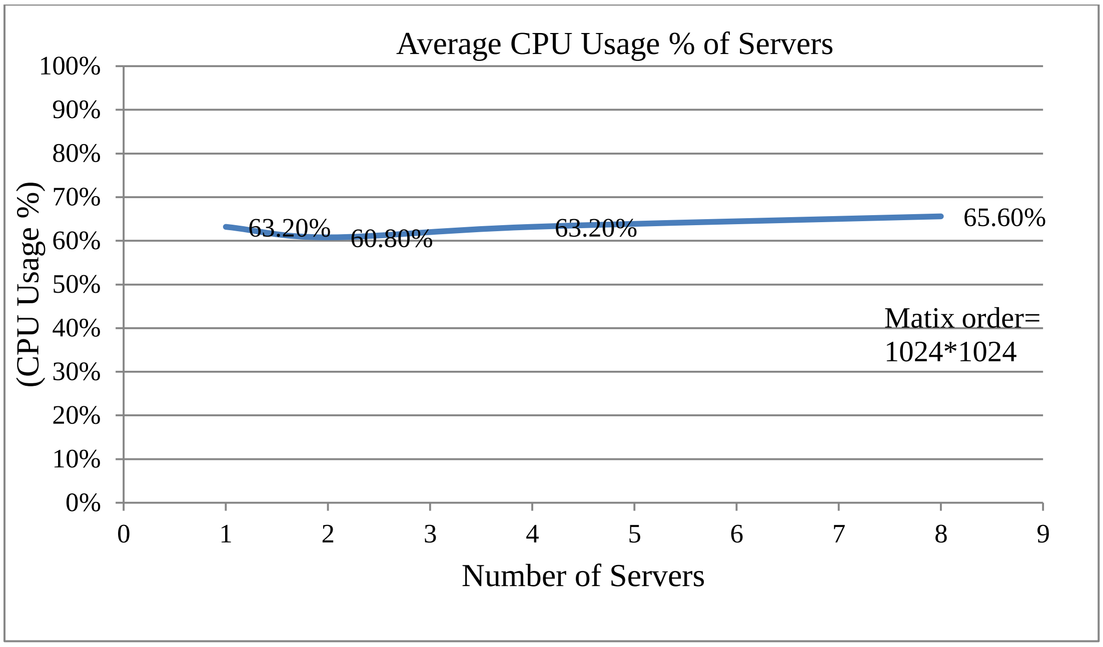


Figure 4 .11. Average CPU Usage % of Servers For Matrix-Algebra of(1024*1024)

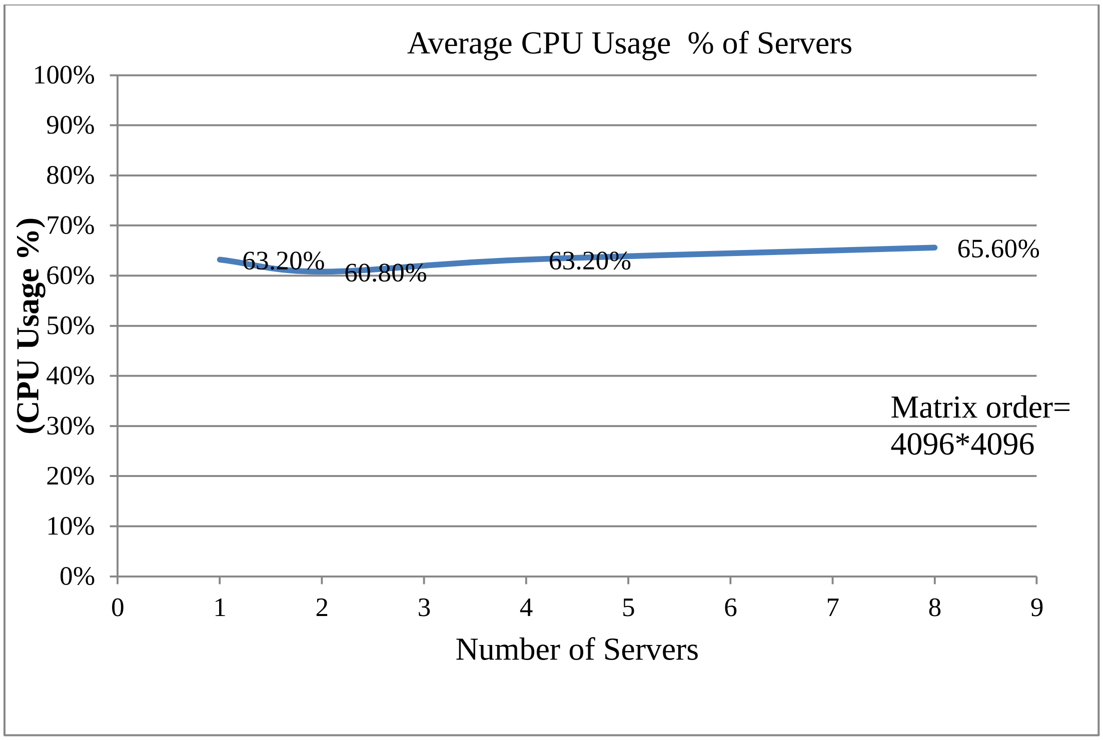


Figure 4.12. Average CPU Usage % of Servers For Matrix-Algebra of(4096*4096)

Table 4.5: Maximum Values of Matrix-Algebra Case-Study For:

(One, Two, Four, and Eight,) Servers.

	Matrix Order = 1024 * 1024			
Time (Sec.)	One server	Two servers	Four servers	Eight servers
Max Consumed CPU time	17.025	10.0875	7.2625	5.325
Max Total Execution Time	21.304184	13.1406	8.82682	6.27662

Table 4.6: Maximum Values of Matrix Algebra-Case For:

(One, Two, Four, and Eight,) Servers.

	Matrix order = 4096 * 4096			
Time (Sec.)	One server	Two servers	Four servers	Eight servers
Max Consumed CPU time	1312.9128	684.212	387.775	235.65
Max Total execution time	1401.3608	757.955	454.114	291.271

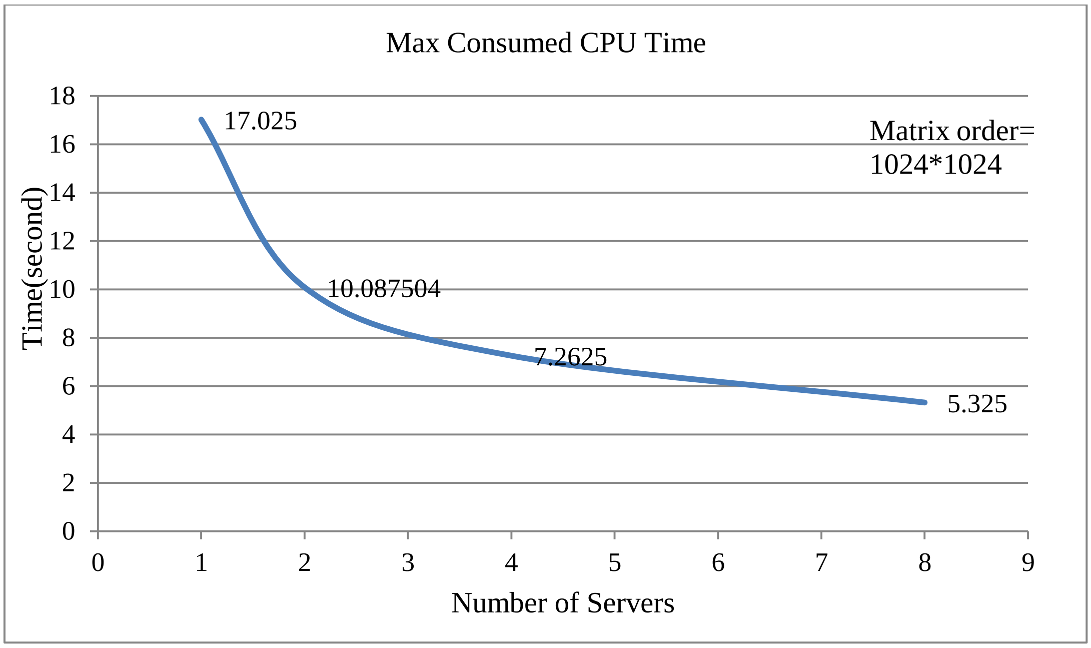


Figure 4.13. Maximum Consumed CPU Time For Matrix-Algebra of (1024*1024)

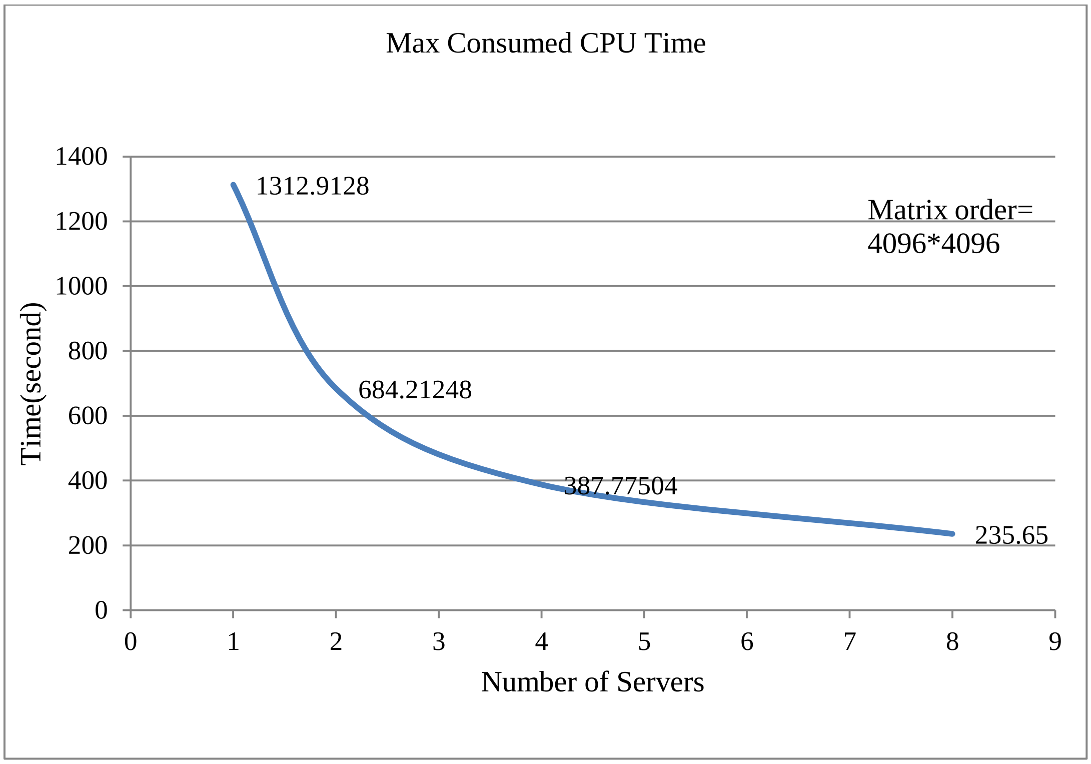


Figure 4.14. Maximum Consumed CPU Time For Matrix-Algebra of (4096*4096)

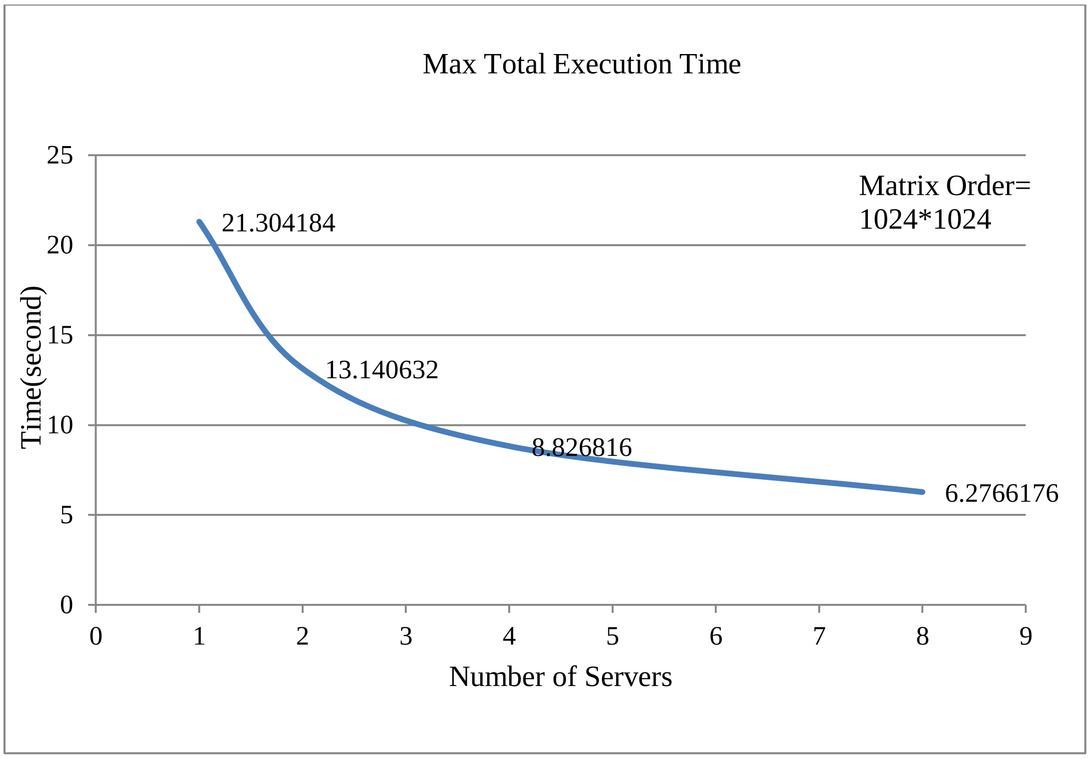


Figure 4.15. Maximum Total Execution Time For Matrix-Algebra (1024*1024)

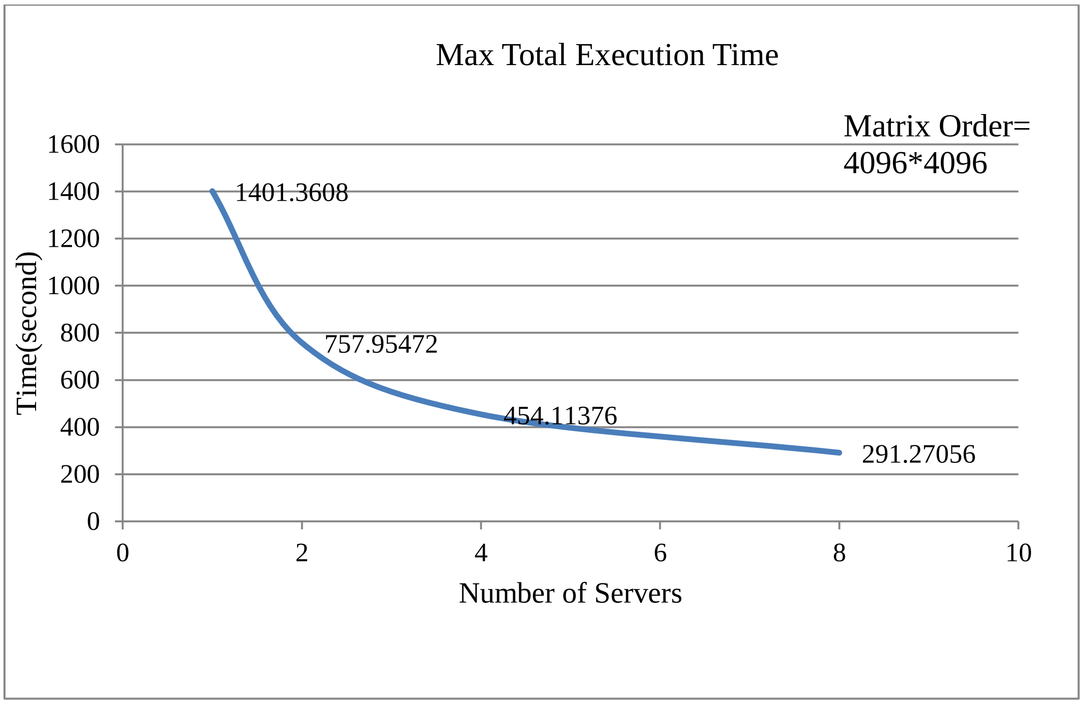


Figure 4.16. Maximum Total Execution Time For Matrix-Algebra (4096*4096)

4.3 Discussion of matrix algebra case-study

As mentioned, there are results of both (Average and Maximum)-values of both of (CPU and total execution)-times.

Figures (4.1 to 4.4) represent four sub-groups of Matrix-orders of results of elapsed Average Servers Consumed CPU time for matrix-algebra operations which are (64, 256 ,1024 and 4096)-orders respectively. This arrangement of curves is dependent because of the high-gap of obtained-results among these four groups. These results are very acceptable. While Figures (4.5 to 4.8) represent four sub-groups of these orders for results of total execution-time of Matrix-algebra operations. It is clear that these results are very acceptable which brows the behavior of the proposed system according to the load-increasing.

Figures (4.13 and 4.14) represent the results of Maximum consumed CPU-time for orders (1024 and 4096) respectively, these results also are very acceptable. The results of orders less than (1024) which are small-loads, are ignored because of the instability of decreasing the Maximum consumed CPU-time with the increasing number of servers. This is applied also on the results of Figures (4.15 and 4.16) that related with Maximum total execution-time of the program.

Figure (4.9, 4.10, 4.11 ,412) illustrates the Average Servers CPU-usage of all servers with all tested sets of Matrix-orders; it represents the relationship between CPU-usage and number of servers to determine the Average Servers CPU-usage according to all cases of matrix-orders. It is clear that CPU-usage is increasing with increasing of the load for the same number of used-servers. It is expected that for each certain number used servers, the value of CPU usage will increase by increasing the load. This is clear with the cases of high-number of servers (i.e. > 2 servers), but for (≤ 2 servers) the average of CPU-usage is unstable and may be the changing is independent on this manner because the value of CPU-usage is affected by any instance under-running tasks depending on the computer status also may be the nature of the data that under-processing effects on the value of CPU-usage.

4.4 Conclusion of obtained results

There is no any previous work depended exactly on the case-studies addressed in this work, because each work applied the principles of parallel processing on a certain application with certain approach (Technique) of parallel processing.

The important of this work illustrated on; applying the principles of parallel processing and forking the jobs among all processors in a specific approach to overcome the complexity of the problem with as possible as minimum duration of time execution. Adding to that, this work is deal with the pure consumed CPU time, which is had not been addressed clearly before by any previous work.

However, to compare the results obtained by this work with those of previous works; it can be seen that all steps of parallel-processing that applied by previous works, are applied here with a suitable approach. And the principles of parallel-processing operations are implemented correctly here. So, this work produces complete CPU-related calculations with very accurate results and specific concepts are added to the parallel processing approach in general and especially to distributed type of them.

CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

5.1 Conclusions

The main points arise from the research employed in this thesis can be summarized by the following:

1. Distributed memory systems addressed depending on client/servers principles with a network consists of nine nodes one of them is a client and the others are servers.
2. Started, Consumed, and Terminated values for (CPU and total execution) times, CPU usage of servers, and (CPU and total execution) times for the Client are calculated by using specific algorithms based on Matrix Algebra case study. There are many general algorithms and other related algorithms which are designed and tested completely by this work.
3. The obtained results are checked and monitored by special programming-checking-subroutines through many testing-iterations and proved a high degree of accuracy.
4. The results showed that parallel-processing operations are ineffective and inefficient with small load applications, and this efficiency is growing with increasing the task load. So, the highest load task will be implemented in high efficiency and the lowest load task implemented with low efficiency, taking in the consideration number of servers used.

5.2 Suggestions for future work

The following are some suggestions for the future works:

1. Apply approaches of monitoring the CPU and states of the processes and threads during implementing the parallel processing approaches. And design software that can control the priority of processes and threads during their life-cycles of implementing the parallel processing approaches.

2. Improve this approach to be applied with for cloud computing via Internet instead of a LAN network.

REFERENCES

[1] **HANK DIETZ , HANKD@ENGR.UKY.EDU (2004)**, Linux Parallel Processing HOWTO, http://aggregate.org/LDP/, v2.0 .

[2] **GEBALI FAYEZ (2011)**, Algorithms and Parallel Computing, Vol. 84. Wiley. com.

[3] **PARHAMI BEHROOZ (2002)**, Introduction to Parallel Processing: Algorithms and Architectures ,Kluwer Academic.

[4] **GARG, K.VIJAY (2004)** ,Concurrent and Distributed Computing in Java, John Wiley & Sons, Inc.

[5] **EL-REWINI, HESHAM, MOSTAFA ABD-EL-BARR (2005)**, Advanced computer architecture and parallel processing , Vol. 42. Wiley. com.

[6] **STALLINGS WILLIAM (2010)** , Computer Organization and Architecture Designing For Performmance 8th Edition, Pearson Education, Inc.

[7] **RAUBER THOMAS, GUDULA RÜNGER (2010)**, Parallel programming: For Multicore And Cluster Systems ,Springer.

[8] **EECKHOUT LIEVEN (2010)**, Computer architecture performance evaluation methods, Synthesis Lectures on Computer Architecture 5.

[9] **LOOSLEY CHRIS , FRANK DOUGLAS (1998)** ,High-performance client/server: a guide to building and managing robust distributed systems , John Wiley & Sons, Inc.

[10] **EDDY WESLEY M. ,MARK ALLMAN(2001)**, Advantages of Parallel Processing and the Effects of Communications Time, Ohio University and NASA GRC / BBN Technologies.

[11] **PAN LEI , ET AL (2002)**, From Distributed Sequential Computing to Distributed Parallel Computing, School of Information and Computer Science University of California, Irvine, CA 92697-3425, USA.

[12] **FRANCOIS, ALEXANDRE RJ(2004)**, Hybrid Architectural Style for Distributed Parallel Processing of Generic Data Streams, University of Southern California Los Angeles, CA.

[13] **SILVA, LUCIANO, DENISE STRINGHINI, ISMAR FRANGO SILVEIRA(2005)**, Parallel Processing Of Distributed Trees: A Pattern Language And Applications, University Mackenzie, Brazil.

[14] **O.YASEEN NUMAN (2010)**, Diagnostic Approach for Improving the Implementation of Parallel Processing Operations, MSc Thesis, University of Zakho, October.

[15] **WANG, QING, ZHENZHOU JI, TAO LIU QING W. (2011)** ,Efficient OpenMP Extension for Embedded Multicore Platform, Journal of Frontiers of Computer Science and Technology, Vol. 5, No. 1.

[16] **FATOHI BAN B. (2011)** ,Modified Approach for Processes and Threads Monitoring and Tracking , MSc Thesis, University of Zakho, Dec.

[17] **ASAAD FARAH H. (2011)** , Shared Memory Performance Analysis on Parallel Processing Applications , MSc Thesis, University of Zakho, Dec.

[18] **RASHID ZRYAN N.(2012)** , Client/Servers Clustering Effects on CPU Execution-Time, CPU Usage and CPU Idle Depending on Activities of Parallel-Processing-Technique Operations, MSc Thesis, University of Sulaimani, Jan.

[19] **SHAHIDEHPOUR MOHAMMAD, YAOYU WANG (2003)**, Communication and Control in Electric Payer Systems: Application of Parallel and Distributed Processing, John Wiley t Sons, Inc.

[20] **TANIAR, DAVID, CLEMENT HC LEUNG, WENNY RAHAYU, SUSHANT GOEL (2008)** ,High Performance Parallel Database Processing And Grid Databases , A John Wiley & Sons, Inc.

[21] **BARNEY BLAISE (2010)**, Introduction To Parallel Computing , Lawrence Livermore National Laboratory 6.1.

[22] **MARINEAU-MES SEBASTIEN (2003)**, Using Symmetric Multiprocessing (SMP) to Scale Data Plane and Control Plane Performance.

[23] **CHAPMAN, BARBARA, GABRIELE JOST, RUUD VAN DER PAS (2008)**, Using OpenMP: portable shared memory parallel programming ,Vol. 10. The MIT Press.

[24] **LIU, ZHAOBIN, WENYU QU, HAITAO LI, MIN RUAN, WANLEI ZHOU (2009)** , I/O scheduling and performance analysis on multi-core platforms, Concurrency and Computation: Practice and Experience 21.10 1405-1417.

[25] **SCHNEIDEWIND NORMAN F. (2012)** , Computer Network Software and Hardware Engineering with Applications, John Wiley & Sons, Inc.

[26] **MCCOOL MICHAEL, JAMES REINDERS, ARCH ROBISON (2012)** , Structured Parallel Programming: Patterns for Efficient Computation , Elsevier, Inc.

[27] **HAGER GEORG, GERHARD WELLEIN (2011)** , Introduction to high performance computing for scientists and engineers , CRC Press.

[28] **ZOLTặN JUHặSZ , PÉTER KACSUK , DIETER KRANZLMULLER (2005)** , Distributed and parallel systems: cluster and grid computing, springer.

[29] **KIRK, DAVID B., W. HWU WEN-MEI (2010)** , Programming Massively Parallel Processors: A Hands-on Approach ,Morgan Kaufmann.

[30] **WITTWER TOBIAS (2006)**, Introduction to Parallel Programming ,Vol. 1. VSSD.

[31] **MORRISON S.RICHARD (2003)** , Cluster Computing-Architectures, Operating Systems, Parallel Processing, & Programming Languages, GNU General Public Licence.

[32] **CULLER DAVID, JASWINDER PAL SINGH, ANOOP GUPTA (1997)** , Parallel Computer Architecture: A Hardware/Software Approach , Morgan Kaufmann Publishers.

[33] **BRUASET, ARE MAGNUS, ASLAK TVEITO (2006)** , Numerical Solution of Partial Differential Equations on Parallel Computers (Lecture Notes in Computational Science and Engineering), Springer-Verlag New York, Inc.

[34] **Q. ZHANG , L. CHENG, R. BOUTABA (2010)** , Algorithms and Architectures for Parallel Processing , Springer-Verlag Berlin Heidelberg 7-18.

[35] **GEIST AL, ADAM BEGUELIN , JACK DONGARRA , WEICHENG JIANG , ROBERT MANCHEK , VAIDY SUNDERAM (1994)** , PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing , MIT press.