

A NEW SCHEDULING TECHNIQUE  
TO IMPROVE DATA MANAGEMENT IN  
CLOUD COMPUTING

NAWSHER KHAN

UMP

DOCTOR OF PHILOSOPHY  
UNIVERSITY MALAYSIA PAHANG

**THESIS CONFIDENTIAL STATUS**  
**UNIVERSITY MALAYSIA PAHANG**

**DECLARATION OF THE THESIS AND COPYRIGHT**

Author's full name : Nawsher Khan

Date of birth : 30/12/1977

Title : A NEW SCHEDULING TECHNIQUE TO IMPROVE  
DATA MANAGEMENT IN CLOUD COMPUTING

Academic Session : 2009-2013

I declared that this thesis is classified as:

- CONFIDENTIAL** (Contains confidential information under the official Secret Act 1972)
- RESTRICTED** (Contains restricted information as specified by the organization where research is done)
- OPEN ACCESS** I agree that my thesis to be published as online open access (Full text)

I acknowledge that Universiti Malaysia Pahang reserves the right as follows:

1. The thesis is the property of Universiti Malaysia Pahang.
2. The library of Universiti Malaysia Pahang has the right to make copies for the purpose of the research only.
3. The library has the right to make copies of the thesis for academic exchange.

Certified by:

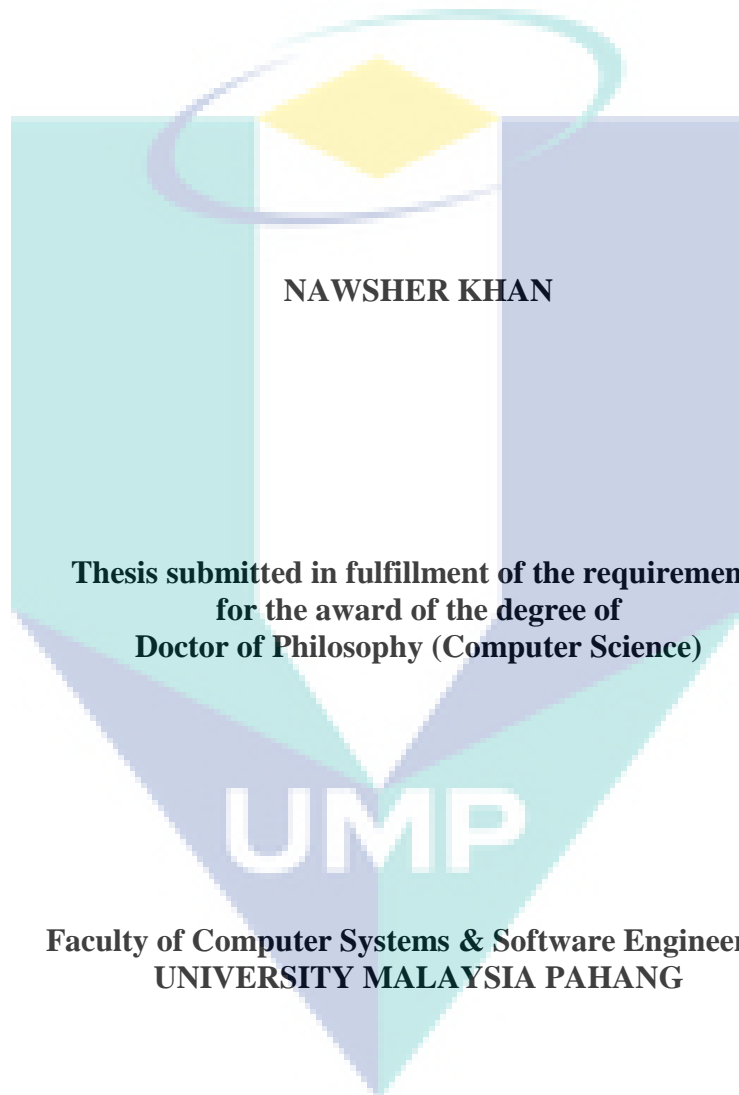
\_\_\_\_\_  
(Student's Signature)

CW4118061  
Passport Number  
Date: April 10, 2013.

\_\_\_\_\_  
(Signature of Supervisor)

Assoc. Prof. Dr. Noraziah Ahmad  
Name of Supervisor  
Date: April 10, 2013.

**A NEW SCHEDULING TECHNIQUE  
TO IMPROVE DATA MANAGEMENT IN  
CLOUD COMPUTING**



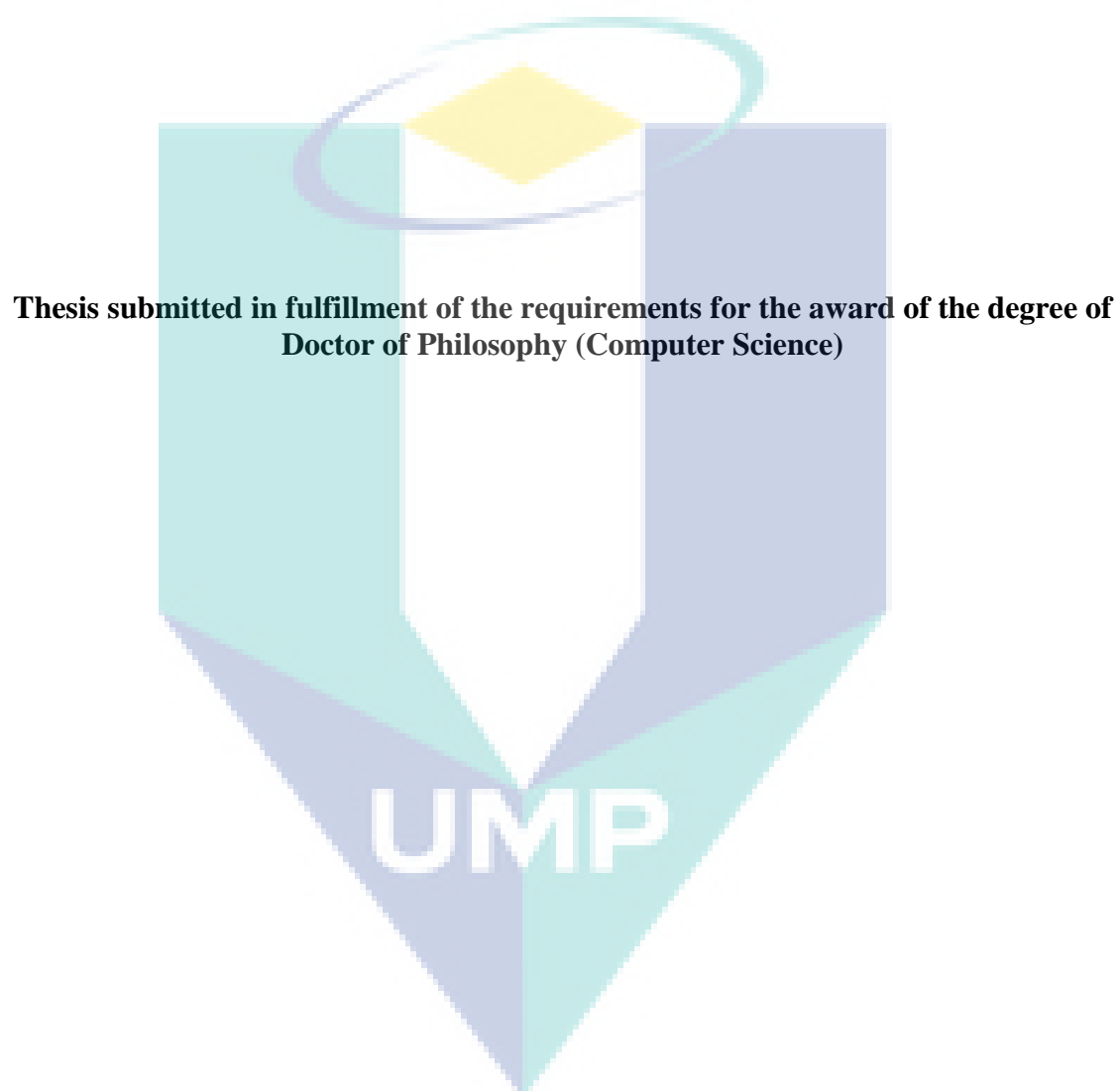
**NAWSHER KHAN**

**Thesis submitted in fulfillment of the requirements  
for the award of the degree of  
Doctor of Philosophy (Computer Science)**

**UMP**

**Faculty of Computer Systems & Software Engineering  
UNIVERSITY MALAYSIA PAHANG**

2013



**Thesis submitted in fulfillment of the requirements for the award of the degree of  
Doctor of Philosophy (Computer Science)**

## SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion this thesis is satisfactory in terms of scope and quality for the award of Doctor of Philosophy in Computer Science.

Signature

:

Name of Supervisor : DR. NORAZIAH BINTI AHMAD

Position : ASSOCIATE PROFESSOR

FACULTY OF COMPUTER SYSTEMS & SOFTWARE  
ENGINEERING, UNIVERSITI MALAYSIA PAHANG

Date

: April 10, 2013.

The logo of Universiti Malaysia Pahang (UMP) is a shield-shaped emblem. It features a central white diamond with a yellow diamond inside it, surrounded by a blue and green circular swirl. The shield is divided into four quadrants: top-left is light blue, top-right is light purple, bottom-left is light purple, and bottom-right is light blue. The letters 'UMP' are written in white at the bottom of the shield.

UMP

## STUDENT'S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

Signature

:

Name

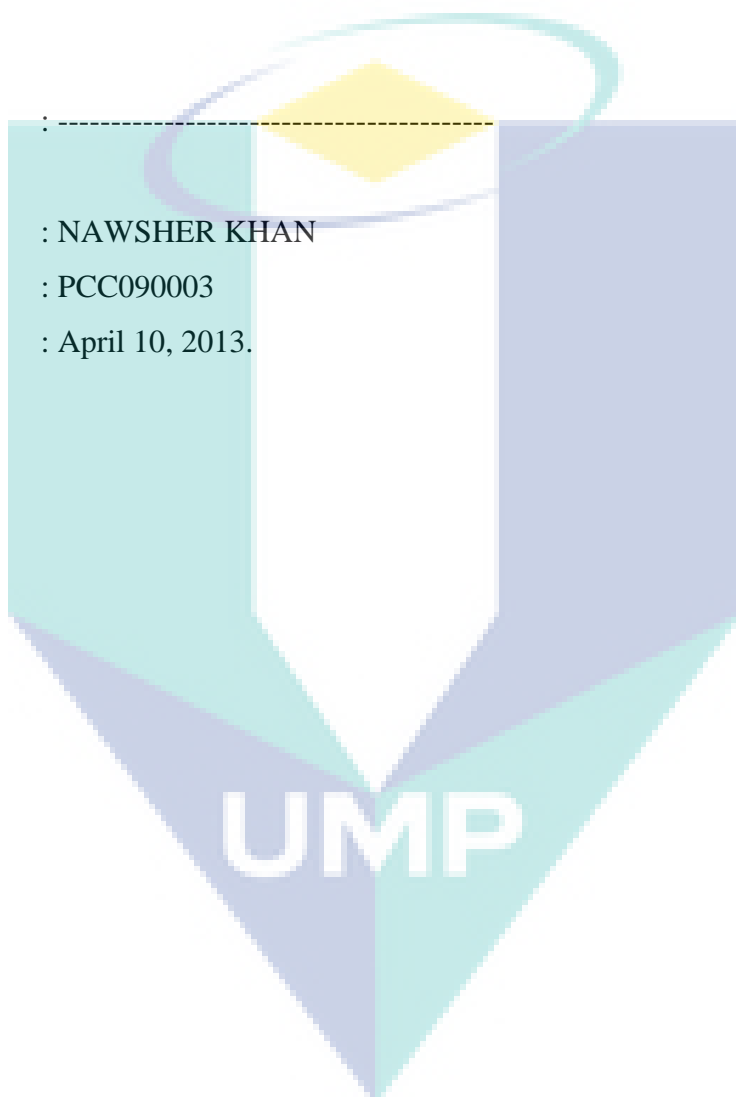
: NAWSHER KHAN

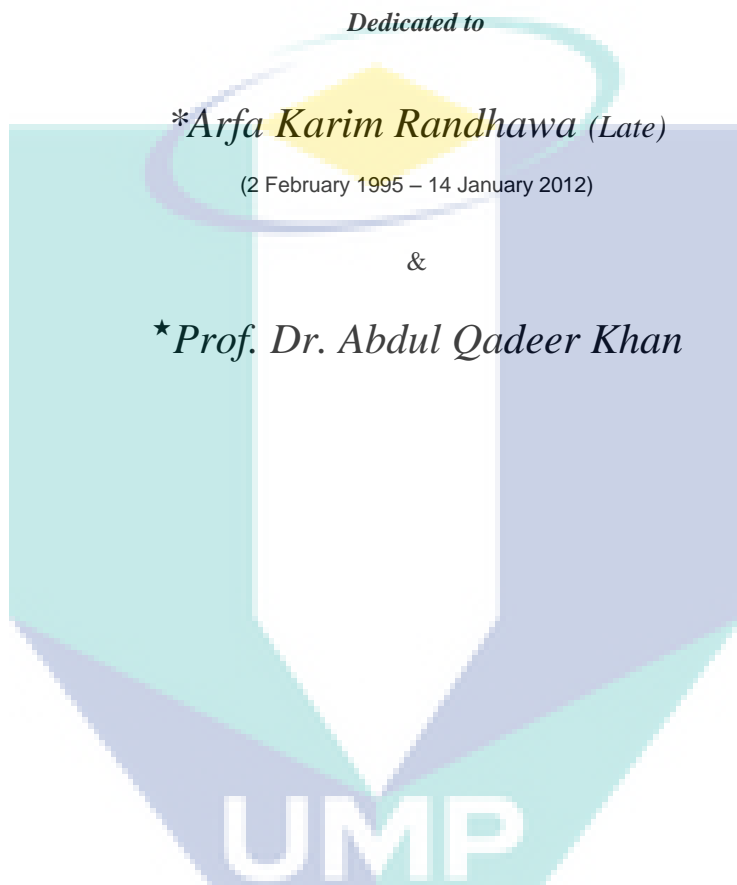
ID Number

: PCC090003

Date

: April 10, 2013.





\* Arfa became the world's youngest Pakistani *Microsoft Certified Professional* (MCP) at the age of nine in 2004. She was invited by Bill Gates to visit the Microsoft Headquarters in the USA. In her life, she was awarded the *Fatimah Jinnah Gold Medal, Salaam Pakistan Youth Award, President's Pride of Performance, the Civil Award* and many more. Further, Arfa was made brand ambassador for Pakistan Telecommunication Company's 3G Wireless Broadband service. She wanted to study at Harvard, work in Microsoft, and explore technological innovations in the field of satellite engineering in Pakistan. She died after 26 days of epileptic seizure. The last job before her death was with NASA.

\* Dr. Abdul Qadeer Khan (A. Q. Khan) is a Pakistani nuclear scientist, metallurgical engineer, and the undisputed hero of Pakistan's nuclear saga and called "*The Father of The Islamic Bomb*". His breakthrough ultimately resulted in the historic explosion of six nuclear bombs in May 1998. Not only was this but a significant development also made with the successful test firing of Intermediate Range Ballistic Missiles, Ghauri 1, in April 1998 and Ghauri II in April 1999. Dr Khan has received honorary degrees of Doctor of Science from the University of Karachi in 1993, Doctor of Science from Baqai Medical University on (1998), Doctor of Science from Hamdard University, Karachi (1999) and Doctor of Science from the University of Engineering and Technology, Lahore in December 2000. For his contributions in the field of science and technology, the President of Pakistan conferred upon Dr Khan the award of Nishan-I-Imtiaz 1996 and 1998. Dr Khan is the only Pakistani to have received the highest civil award of "Nishan-I-Imtiaz" twice. He is also a recipient of Hilal-I-Imtiaz.

## ACKNOWLEDGEMENT

All praises to the Almighty Allah (s.w.t), who made me able to explore new horizon of knowledge. With due reverence, I would like to acknowledge that without the divine blessing, I would not have been able to be part of a well versed family and friends. These factors made me achieve what was destined for me.

I would like to express my sincere gratitude and thanks to my respectable supervisor, Assoc. Prof. Dr. Noraziah Ahmad, who has always guided me through her constructive approach of my endeavors in a sister figured manner. Her assistance, devotion, and encouragement are unforgettable in my life. I am also thankful to my co-supervisor, Dr. Tutut Herawan, who always pushed me to finish my work on time and to the point. I also would like to pay thanks to Prof. Dr. Mustafa Mat Deris from University Tun Hussein Onn Malaysia, for his supervision, who was always very kind to extend his valuable guidance and insight in my study. I also recognize the help and support by the clerical staff in the Faculty of Computer Systems and Software Engineering throughout the duration of my research.

Finally, I would like to appreciate my parents for giving me the best chance, advice, and support for receiving education and special thanks to my wife, who shouldered the whole responsibility of my family and of my kids, in my absence. My gratitude is extended to all my friends, my fellows, especially to my dearest, who always give courage and showed me, “Thumb Up”.

The logo of Universiti Malaysia Perlis (UMP) is a large, stylized letter 'V' shape. It is composed of several overlapping triangles in shades of teal, light blue, and yellow. The letters 'UMP' are written in a bold, white, sans-serif font across the center of the 'V' shape.

UMP



## ABSTRACT

Cloud Computing is an extremely successful service oriented computing paradigm and has revolutionized, modernized, and well-developed infrastructure of computing. It is being signaled as the next-generation shift which combines the Internet and computing, as a result, users will be able to access and store software, content, and data in remote servers run by other companies or by a client. Data Management is the key factor of Cloud Computing, which is '*the right data in the right place at the right time*'. It is also the development and execution of architectures, policies, practices, and procedures in order to manage the information lifecycle needs of an organization in an effective manner. Scheduling techniques, which are the important part of data management, are disciplines and procedures used for distributing resources between two different parties, that is, Cloud Computing provider and Cloud Computing service user. The main purposes of scheduling algorithms, architectures, and techniques are to minimize the starvation of resources and service during the right time for using.

Existing models presents the whole scheduling architecture for data transferring process, by taking in two slots. *External Scheduler (ES)* in one, and *Local Scheduler (LS)* with *Data Scheduler (DS)* in another slot. But new proposed scheduling architecture takes all three scheduler separately. On the base of these three separate schedulers *Queue Time (QT)*, *Execution Time (ET)*, and *Data Transfer time (DT)*, also have been taken separately in data transfer time calculation.

Dealing with increasing huge amount of data makes the requirement more critical for efficient accessing of data. Scheduling techniques have their major involvement in managing day-by-day increased large data in cloud environment. This research proposes a new scheduling technique to calculate the *Total Completion Time (TCT)* for the transfer of specific amount of data. The formula for transfer time calculation has three parameters, namely the Queue Time (*QT*), Execution Time (*ET*), and Data Transfer time (*DT*). All these *times* (intervals) are different from each other and have their own importance during calculation. In previous exist models, one of these values, either *QT* or *DT*, has been ignored by taking maximum of them. Ignoring one value means decreasing the actual consuming time. The proposed model considers each parameter separately, means giving importance to each parameter. As an outcome, the Total Transferring Time for data can be the sum of *QT*, *ET* and *DT* in *TCT*.

The proposed model *Total Completion Time (TCT)* has been evaluated by using a single server and finite population M/M/C/\*P queuing model. There is a great impact on accuracy by taking each parameter separately in the formula. Accuracy is 85% by using 56Kbps bandwidth (*BW*) and number of jobs (*M*) taken 2, it is increased up to 92.4639% for 50 jobs. The accuracy is 98.5000%; for 2 jobs, increases up to 99.1753% for jobs 50 by using BW 512kbps. Result shows that by using  $M > 500$ , stability point (where accuracy is 100%) can be achieved. Hence new technique is more efficient when we need to transfer large amount of data. Experiments showed that the proposed model is more reliable, in terms of accuracy. The proposed model has an accurate transfer time calculation, thus Cloud Computing can present its services in a more efficient manner.

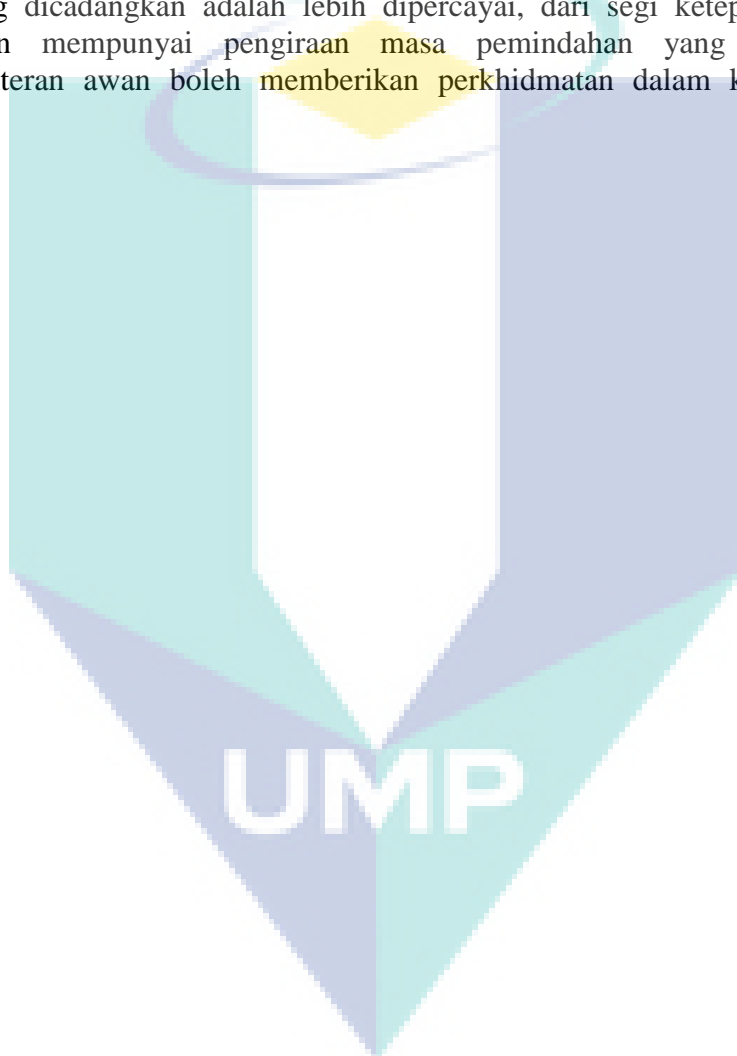
## ABSTRAK

Perkomputeran Awan adalah perkomputeran berasaskan servis yang sangat berjaya. Ianya telah berevolusi, mengalami proses permodenan serta mempunyai infrastruktur yang terancang. Perkomputeran Awan merupakan pengkomputeraan generasi seterusnya dan telah menyebabkan gelombang peralihan. Ini adalah kerana pengkomputeran Awan menggabungkan internet dan perkomputeran yang hasilnya membolehkan pengguna mengakses dan menyimpan perisian, kandungan dan data di dalam pelayan yang disediakan oleh syarikat lain atau pengguna. Pengurusan data adalah kunci utama kepada pengkomputeran Awan di mana 'Data yang tepat pada masa yang tepat'. Pengurusan data adalah pembangunan dan pelaksanaan dari segi rekabentuk, polisi, amalan dan prosedur di dalam pengurusan kitaran maklumat yang diperlukan oleh organisasi dalam cara yang efektif. Teknik penjadualan yang merupakan bahagian yang sangat penting di dalam pengurusan data, merupakan disiplin atau prosedur yang digunakan dalam pem bahagian sumber diantara dua pihak iaitu, penyediaan perkomputeran awan dan pengguna perkhidmatan perkomputeran awan. Tujuan utama algoritma penjadualan, senibina dan teknikal adalah untuk meminimalkan kekurangan sumber dan servis pada masa penggunaan.

Model sedia membentangkan, Mekanisme keseluruhan senibina penjadualan dengan mengambil kira dua slot. *Penjadualan Luar (ES)* dalam satu slot, dan *Penjadualan Setempat (LS)* dengan *Penjadualan Data (DS)* dalam slot yang lagi satu. Walaubagaimanapun senibina penjadualan baru yang dicadangkan mengambil ketiga-tiga penjadualan secara berasingan. Pada asasnya ketiga-tiga penjadualan berasingan *Masa Giliran (QT)*, *Masa Pelaksanaan (ET)*, dan masa *Pemindahan Data (DT)*, juga telah diambil secara berasingan dalam data pengiraan masa pemindahan.

Berurusan dengan peningkatan data yang semakin banyak menyebabkan keperluan yang sangat kritikal kepada capaian data yang lebih efisien. Teknik penjadualan memang banyak terlibat di dalam peningkatan data yang besar hari demi hari di dalam persekitaran awan. Penyelidikan ini mencadangkan teknik penjadualan yang baru untuk mengira *Jumlah Masa Tamat (TCT)* untuk proses penghantaran data yang tertentu. Formula untuk masa penghantaran mempunyai tiga parameter, iaitu Masa Giliran (*QT*), Masa pelaksanaan (*ET*) dan Masa Penghantaran (*DT*). Kesemua jenis masa ini (selang) adalah berbeza antara satu sama lain dan mempunyai nilai kepentingan yang tersendiri semasa pengiraan. Di dalam model yang sebelumnya, salah satu nilai *QT* atau *DT* adalah diabaikan kerana mereka hanya mengambil kira nilai yang maksimum. Mengabaikan salah satu nilai bermakna terdapat penurunan kepada nilai masa yang sebenarnya diperlukan. Model yang dicadangkan ini menitik beratkan setiap parameter di dalam formula yang sedia ada secara berasingan iaitu dengan memberikan kepentingan untuk setiap parameter. Hasilnya, Jumlah Masa Penghantaran adalah jumlah kepada *QT*, *ET*, *DT* dan *TCT*.

Model yang dicadangkan *Jumlah Masa Tamat (TCT)* telah dinilai dengan menggunakan pelayan tunggal dan populasi terhingga model giliran  $M / M / C / * / P$ . Terdapat kesan yang besar terhadap ketepatan dengan mengambil setiap parameter secara berasingan dalam formula. Ketepatan adalah 85% dengan menggunakan *jalur lebar(BW)* 56kbps dan bilangan tugas ( $M$ ) 2, ianya meningkat sehingga 92.4639% untuk 50 tugas. Ketepatan adalah 98.5000%; untuk 2 tugas dan peningkatan sehingga 99.1753% untuk 50 tugas dengan menggunakan BW 512kbps. Keputusan menunjukkan bahawa dengan menggunakan  $M > 500$ , titik kestabilan (di mana ketepatan adalah 100%) boleh dicapai. Oleh itu teknik baru yang dicadangkan adalah lebih cekap apabila kita perlu untuk memindahkan sejumlah data yang besar. Eksperimen menunjukkan bahawa model yang dicadangkan adalah lebih dipercayai, dari segi ketepatan. Model yang dicadangkan mempunyai pengiraan masa pemindahan yang tepa tseterusnya pengkomputeran awan boleh memberikan perkhidmatan dalam kaedah yang lebih cekap.



## TABLE OF CONTENTS

	<b>Page</b>
<b>SUPERVISOR’S DECLARATION</b>	v
<b>STUDENT’S DECLARATION</b>	vi
<b>DEDICATION</b>	vii
<b>ACKNOWLEDGEMENTS</b>	viii
<b>ABSTRACT</b>	ix
<b>ABSTRAK</b>	x
<b>TABLE OF CONTENTS</b>	xii
<b>LIST OF TABLES</b>	xvi
<b>LIST OF FIGURES</b>	xvii
<b>LIST OF ABBREVIATIONS</b>	xix
<b>CHAPTER 1      INTRODUCTION</b>	
1.1      Introduction	1
1.2      Cloud Computing	1
1.3      Data Management	3
1.4      Scheduling	3
1.5      Problem Statement	4
1.6      Objectives of Research	8
1.7      Scope of Research	8
1.8      Structure of the Thesis	8
<b>CHAPTER 2      LITERATURE REVIEW</b>	
2.1      Introduction	11
2.2      Grid Computing	11
2.3      Cloud Computing	12
2.3.1    Major Benefits	14
2.3.2    Cloud comparison with Cluster and Grid	16
2.3.3    Types of Clouds	19

	2.3.3.1	Infrastructure-as-a-Service (IaaS)	20
	2.3.3.2	Platform-as-a-Service (PaaS)	21
		2.3.3.2.1 PaaS Features	22
	2.3.3.3	Open Source Cloud	25
		2.2.3.3.1 Open Source Platforms Comparison	26
	2.3.3.4	Software-as-a-Service (SaaS)	27
	2.3.3.5	Data-as-a-Service (DaaS)	29
	2.2.4	Cloud Prices	30
2.4		Data Management In Cloud Computing	31
	2.4.1	Scheduling	32
		2.4.1.1 Scheduling Architectures	33
		2.3.1.2 Job Scheduling Architectures Summery	40
	2.4.2	Queuing Theory	41
		2.4.2.1 Kendall's Classification	43
	2.4.3	Queuing Models	44
		2.4.3.1 M/M/1 Model	44
		2.4.3.2 M/M/1/K Model (Capacity Constraint)	45
		2.4.3.3 M/M/c Model (Multi-server System)	46
		2.4.3.4 M/M/c/K Model (Capacity Constraint with Multi-Server System)	46
		2.4.3.5 M/M/c/*/P Model (Finite Population)	46
	2.4.4	Queue Characterization	49
2.5		Summary	51

## CHAPTER 3 METHODOLOGY

3.1	Introduction	53
3.2	Queuing Theory And Scheduling	53
	3.2.1 Parameters Used In Queuing Theory	54
	3.2.1.1 Independent Parameter	54
	3.2.1.2 Dependent Parameter	55

3.2.1.3	Parameters Definition and Explanation	55
3.2.2	Scheduler Architecture	59
3.2.3	Scheduling Algorithm for TCT	63
3.2.3.1	ALGORITHM'S DESCRIPTION	64
3.2.4	Scheduling Strategy	65
3.2.5	Physical Structure of Cloud Architecture	66
3.2.5.1.	Application Programming Interface	67
3.2.5.2.	Cloud Administration	67
3.2.5.3.	Data-as-a-Services	68
3.2.5.4.	Cloud Schedulers	68
3.2.5.5.	Virtual Infrastructure Management (VIM)	69
3.2.5.6.	Cloud Networks	69
3.2.6	Queuing System	70
3.2.6.1	M/M/c/*P	70
3.2.7	Proposed Technique	74
3.3	Summary	74
<b>CHAPTER 4 RESULTS AND DISCUSSION</b>		
4.1	Introduction	76
4.2	Parameters Used	77
4.3	M/M/c/*P Model	77
4.4	Queuing Theory Calculator	78
4.5	Transfer Time Calculation	80
4.6	Analysis and Results	83
4.7	Variation of TT and TCT with Bandwidth Change	86
4.7.1	Bandwidth Variation Effect	87
4.8	Variation in TT And TCT with Change in Servers' Number	90
4.9	TT and TCT variation by changing the servers' number with bandwidth	93
4.10	Error and Accuracy Estimation between TT and TCT	102
4.11	Overall Overview	105

4.12	Summary	106
------	---------	-----

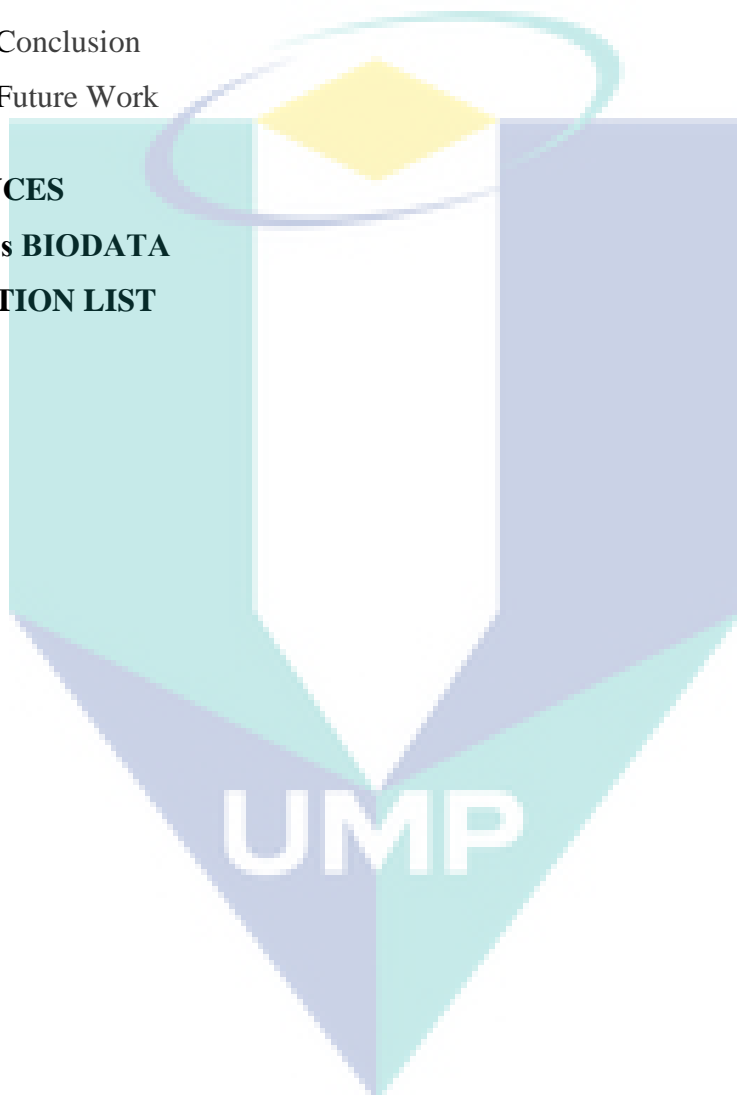
## **CHAPTER 5 CONCLUSION AND FUTURE WORK**

5.1	Introduction	107
5.2	Scientific & Technological Contributions	107
5.3	Limitations	108
5.4	Conclusion	108
5.5	Future Work	109

<b>REFERENCES</b>	111
-------------------	-----

<b>AUTHOR'S BIODATA</b>	116
-------------------------	-----

<b>PUBLICATION LIST</b>	117
-------------------------	-----



**LIST OF TABLES**

<b>Table No.</b>	<b>Title</b>	<b>Page</b>
2.1	Characteristics Comparison of Cluster, Grid and Cloud Computing	17
2.2	Comparison of various platforms	24
2.3	Comparison of Open-Source Cloud Platforms	25
2.4	Comparing Platforms with Implementation Aspects	26
2.5	Cost Comparison for Some Platforms	31
2.6	Scheduling Algorithms Comparison	33
4.1	Comparison of Various Aspects for Queuing Process	85
4.2	Error Estimation with $C = 1$ , $BW = 56\text{Kbps}$	102
4.3	Accuracy Estimation with $C = 1$ , $BW = 512\text{Kbps}$	104
4.4	Overall Overview	105

The image features a large, semi-transparent watermark logo for UMP (Universitas Mitra Widyacarya). The logo is a shield-like shape composed of several overlapping triangles in shades of teal, light blue, and yellow. The letters 'UMP' are prominently displayed in white, bold, sans-serif font across the center of the shield. The watermark is positioned behind the table of contents, partially obscuring the text in the middle rows.



## LIST OF FIGURES


<b>Figure No.</b>	<b>Title</b>	<b>Page</b>
1.1	Non-Exhaustive View of Cloud System	2
1.2	The Phases of Data transferring process	5
1.3	Schedulers Architectures	6
2.1	Resource Provisioning Models	11
2.2	Cloud Computing Simple Architecture	13
2.3	Global Cloud Exchange Infrastructure	16
2.4	Clouds Type	19
2.5	Cloud Services Structure	21
2.6	Cloud Computing Taxonomy Map	23
2.7	Cloud Computing Taxonomy	28
2.8	Job Scheduling Architecture A	34
2.9	Jobs Scheduling Architecture B	35
2.10	Jobs Scheduling Architecture C	37
2.11	Cloud System Structure	38
2.12	Cloud Task Scheduling Process	39
2.13	M/M/1 State Transition Diagram	45
2.14	M/M/K Model State Transition Diagram	45
2.15	Schematic Representation of A Queuing System	49
3.1	Time in Queue	55
3.2	Time in Server	56
3.3	Schematic representation of time in system	56
3.4	Time in System	57
3.5	Transferring Time	57
3.6	Total Completion Time for Data Transfer	58
3.7	Scheduling Architecture	59
3.8	Physical Structure of Cloud Architecture	66
3.9	Arrivals and Departures from a System	72

4.1	Queuing Theory Calculator	78
4.2	Transfer Time Calculator	81
4.3	General Comparison of TT and TCT	84
4.4	TT and TCT with 56Kbps	86
4.5	TT and TCT with 128Kbps BW	87
4.6	TT and TCT with 256Kbps BW	88
4.7	TT and TCT with 512Kbps BW	89
4.8	TT and TCT using single server	90
4.9	TT and TCT using two servers	91
4.10	TT and TCT using three servers	92
4.11	TT and TCT by a single server with 128Kbps BW	93
4.12	TT and TCT by two servers with 128Kbps BW	94
4.13	TT and TCT by three servers with 128Kbps BW	95
4.14	TT and TCT by a single server with 256Kbps BW	96
4.15	TT and TCT by two servers with 256Kbps BW	97
4.16	TT and TCT by three servers with 256Kbps BW	98
4.17	TT and TCT variation with $C = 1$ , $BW = 512Kbps$	99
4.18	TT and TCT variation with $C = 2$ , $BW = 512Kbps$	100
4.19	TT and TCT variation with $C = 3$ , $BW = 512Kbps$	101
4.20	Error Estimation with $C = 1$ , $BW = 512Kbps$	103

The logo for UMP (Universitas Muhammadiyah Palembang) is a large, stylized shield shape. It is divided into four quadrants by a vertical and a horizontal line. The top-left quadrant is light blue, the top-right is light green, the bottom-left is light purple, and the bottom-right is light teal. The letters 'UMP' are written in a bold, white, sans-serif font across the center of the shield.

UMP

## LIST OF ABBREVIATIONS



AaaS	Architecture-as-a-Service
API	Application Programming Interfaces
AWS	Amazon Web Services
BW	BandWidth
CAPEX	CAPital EXpenditure
CC	Cloud Computing
CPU	Central Processing Unit
DaaS	Data-as-a-Service
DDBMS	Distributed Database Management System
DDS	Distribute Data System
DDS	Distributed Database System
Docs	Documents
DS	Data Scheduler
EC2	Elastic Compute Cloud
ES	External Scheduler
FaaS	Framework-as-a-Service
FCFS	First-Come-First-Served
FIFO	First-In-First-Out
FPPS	Fixed-Priority-Pre-emptive-Scheduling
GIG	Global Information Grid
GS	Grid Structure
GUI	Graphical User Interface
HaaS	Hardware-as-a-Service
HRS	Hierarchical Replication Scheme
I/O	Input/Output
I/P/SaaS	Infrastructure/Platform/Software-as-a-Service
LAN	Local Area Network
LCFS	Last Come First Served
LS	Local Scheduler
NaaS	Network-as-a-Service

OPEX	OPERational EXpenditure
OS	Operating System
PaaS	Platform-as-a-Service
PS	Processing Sharing
QoS	Quality of Service
RaaS	Recovery-as-a-Service
RAM	Random Access Memory
ROI	Return of Investment
MSTS	Monitoring Synchronization Transactions Systems
RR	Round-Robin
SaaS	Software-as-a-Service
SIRO	Service In Random Order
SJF	Similar to Shortest First
SLA	Service Level Agreement
SMB	Small and Medium Business
SOA	Service Oriented Architecture
SPT	Shortest Processing Time
SST	Shortest Service Time
TCT	Total Completion Time for transferring
TE	Time in Execution
TQ	Tree Quorum
TQ	Time in Queue
TS	Time in System
VaaS	Voice-as-a-Service
VIM	Virtual Infrastructure Management
VM	Virtual Machine
WAN	Wide Area Network
WS	Web Services

## CHAPTER 1

### INTRODUCTION

#### 1.1 INTRODUCTION

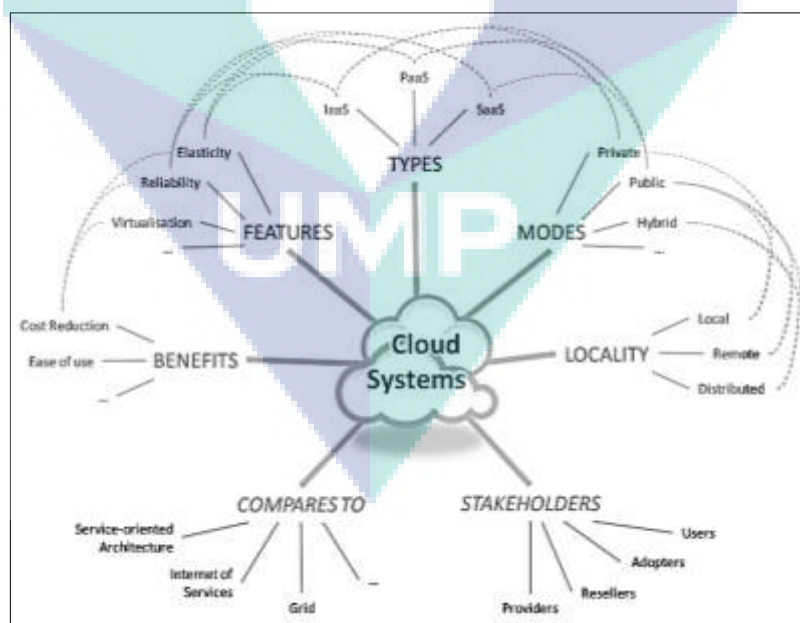
Scheduling is the key factor of Data Management in any environment, especially in Distributed Systems, Grid Computing and Cloud Computing. Virtualized manner platforms, storage, computing power, different services and application are presenting with a nice manner to external jobs over the Internet. *Right data in right place at the right time* is the efficient data management, which is the basic need for Cloud Computing services provision. Efficient provision of the service depends on the well-organized data management system, and data management system depends on the reliable scheduling techniques with accurate data transportation time calculation in cloud environment.

#### 1.2 CLOUD COMPUTING

Cloud Computing is a tremendous revolution for the provision of various services. The vision of Cloud Computing and Grid Computing is the same: to minimize the computing charges and to increase reliability as well as elasticity. Now things have changed, as compared with the situation ten years ago. The idea of Cloud Computing is not new, as stated by Mc. Carthy in 1969, "Computation may someday be organized as a public utility and let see how this speculates might occur" (Kleinrock, 2005). However, with implementation aspects, Cloud Computing is very new. (Buyya et al.,

2011) proclaimed that, “Cloud Computing is an utility computing model for on-demand delivery of computing power; consumers pay providers based on usage (‘pay-as-you-go’), similar to the way in which we currently obtain services from traditional public utility services such as water, electricity, gas, and telephony” (Buyya et al., 2011).

Cloud Computing, as defined by (Vaquero et al., 2009; and Weiss, 2007), is the manner where tremendously scalable IT enabled capabilities, and utilities are delivered *as-a-service* to external jobs using internet technologies. The Cloud offers so many benefits, as shown in Figure 1.1, that is, fast deployment, rapid provisioning, pay-for-use, scalability, lower costs, fast elasticity, greater resiliency, ubiquitous network access, on demand security controls, hypervisor (virtual machine) protection against network attacks, low-cost disaster recovery and data-storage solutions, real time detection of system, and tampering with rapid reconstitution of services. Data management is the base of Cloud Computing, the function of which is to deliver various services efficiently.



**Figure 1.1:** Non-exhaustive view of the Cloud System

Source: Say People 2012

### 1.3 DATA MANAGEMENT

Data-intensive and high-performance computing applications require efficient management and transfer, and then handle terabytes or peta bytes of information in distributed computing environments. Users need to be able to transfer large subsets of datasets to local sites or other remote resources (Aggregate, 2012). Data management is the improvement and execution of architectures, practices, policies, and procedures in order to organize the informative life cycle needs of an enter-prise Cloud services in an effective manner. In well-organized management policies and protocol, the scheduling plays an important and active role.

Data Management is an important aspect particularly in storing Clouds, where data is flexibly distributed across multiple resources. Implicitly, data consistency needs to be maintained over a wide distribution of replicated data sources. At the same time, the system always needs to be aware of the data location during replication. For taking decision of *when* and *where* to execute data, Cloud providers need efficient scheduling techniques which is the key factor in data management, to deliver services with effective scalability and reliability.

### 1.4 SCHEDULING

Scheduling disciplines are procedures used for distributing resources between two different parties (Cloud provider and Cloud user) which simultaneously and asynchronously request them. The main purposes of scheduling algorithms, architectures, and techniques are to minimize the starvation of resources and service during the right time using for data transfer (Shi et al., 2010). Effective scheduling disciplines and algorithms are needed to ensure fairness among the provider and the user for the utilization of resources and cloud services. Scheduling deals with the problems of deciding which of the outstanding requests can be allocated resources (Deelman and Chervenak, 2008). With virtualized manner platforms, computing power, storage, and different services are delivered on demand to external jobs over the

Internet. Hence *right data in the right place at the right time* is efficient data management, which is the basic need for Cloud Computing services provision and scheduling is the main factor of data management.

There are many algorithms in scheduling. One of them is First-Come-First-Served (FCFS), which is the simplest scheduling algorithm. Similar to Shortest-Job-First (SJF), this policy allows the scheduler to arrange processes with the minimum estimated processing time remaining to be next in the queue. In Fixed-Priority-Pre-emptive-Scheduling (FPPS), each process is getting a fixed priority rank from the operating system, and in the ready queue, according to their priority order, the scheduler arranges the processes. Lower priority processes get interruption by incoming higher-priority processes. Round-Robin-Scheduling (RRS) assigns a fixed time unit per process, and cycles through them. Multilevel-Queue-Scheduling (MQS) can be used for situations in which processes are easily segmented into different groups.

## 1.5 PROBLEM STATEMENT

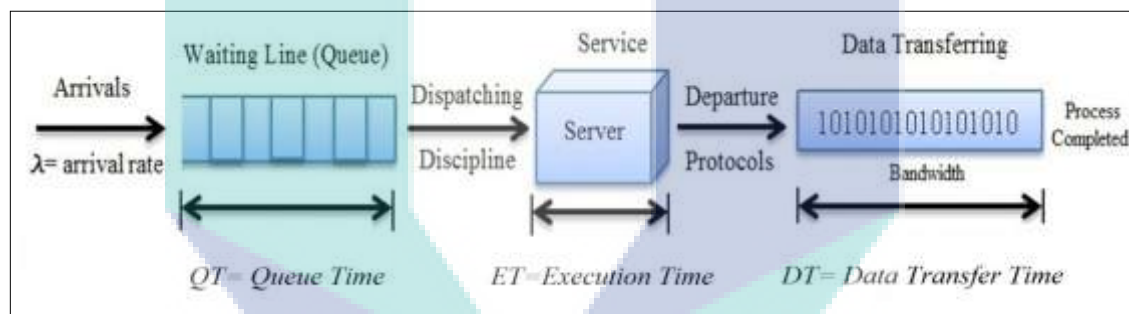
Scheduling technique (Equation 1.1) has proposed by (Tang et al., 2006; Ranganathan and Foster, 2003; and Nguyen and Lim, 2007). The following technique  $TT$  for Total Transfer Time calculation for a job has been used.

$$TT_{k,i} = \max\{QT_{(i)}, DT(f(k), i)\} + ET_{k,i} \quad (1.1)$$

Where  $TT$  = Total Transfer Time,  $QT$  = Queuing Time;  $DT$  = Data Transfer Time; and  $ET$  = Job Execution Time. Only one value  $DT$  or  $QT$  has been considered in  $\max\{QT(i), DT(f(j), i)\}$  in Equation 1.1 because only one value out of these two will be maximum, the other minimum  $DT$  or  $QT$  value has been ignored.  $QT$  and  $DT$  both are two different parameters, and have their own importance separately.



There are three ( $QT$ ,  $ET$  and  $DT$ ) parameters in Equation 1.1.  $QT$  is the time which a job passes in the queue before starting the execution. Time denoted by  $ET$  is the job execution time before starting the transferring process.  $DT$  is the time after completing the execution time and before the completion data transferring process.  $TT$  is the total time for transfer completion, and  $ETTC$  is the estimated total time for completion transferring. Here in  $\max\{QT_{(i)}, DT_{(f(j),i)}\}$  the value of either  $QT$  or  $DT$  has been ignored, by taking the maximum of both, by (Tang et al., 2006; Ranganathan and Foster, 2003; and Nguyen and Lim, 2007). Ignoring one value means decreasing the actual consuming time. According to experiments, almost  $DT$  is ignored because  $DT$  is minimum value (Table 4.1, Table 4.2, Table 4.3 and Table 4.4), as compared to  $QT$ . All these three parameters ( $QT$ ,  $ET$  and  $DT$ ) are different from each other, and each one has its own importance respectively as shown in Figure 1.2.

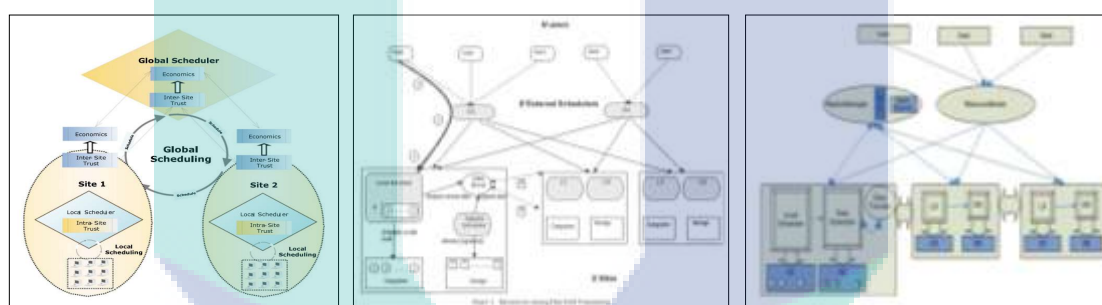


**Figure 1.2:** The Phases of Data transferring process

This study presents an alternative proposed scheduling technique, which give importance to each parameter by considering each one separately in the existing formula while calculating the Total Completion Time ( $TCT$ ). As an outcome, the Total Transferring Time for data can be the sum of *Queue Time*, *Execution Time*, and *Data Transfer time*.

Job scheduling architectures A (Nguyen and Lim, 2007), B (Ranganathan and Foster, 2003) and C (Liang and Shi, 2010) basically have divided their schedulers into

three phases. For the job submission by the users all three architectures follow First-Come-First-Serve (FCFS) scheduling policy. User jobs are submitted directly to *ES* (External Scheduler) and *RB* (Resource Broker) in both architectures where jobs are placed in queues. Both *ES* and *RB* have different scheduling policies at this level for job dispatch to *LS*. *ES* uses (i) load at the remote site and/or (ii) location of dataset as the resource scheduling strategy whereas *RB* uses estimation of cost in terms of time as the resource scheduling strategy. In Job scheduling architectures C, the architecture is also divided into three phases. But instead of *LS* and *DS* like in architecture A and B it presents two *LS* with the *GS* (Global Scheduler). User jobs are submitted and queued to *GS*. *GS* uses stock of currency between the submission site and selected running sites based on the reputation information and resource request. *LS* works by using processor selection based on the intra-site trust information.



Liang and Shi, 2010

Nguyen and Lim, 2007

Ranganathan and Foster, 2003

**Figure 1.3:** Scheduler Architectures

Cheng (2007) proposed two schedulers MWTP and VWTP to perform proportional delay differentiation. Algorithms can maintain the delay proportion and reduce the average queuing delay by simultaneously considering the packet waiting time and the packet transmission time. (Liang and Shi, 2010) proposed a reputation-based resource scheduler for the Grid. (Dwekat and Rouskas, 2011) presented tiered-service fair queuing (TSFQ) scheduler techniques, within each tier, the schedulers employ a fixed number of queues to handle packets with few or no sorting operations. (Francini et al., 2001) have presented three enhancements of WRR schedulers for providing bandwidth guarantees in IP networks.

All above mentioned algorithms have taken in three parts, (Nguyen and Lim, 2007; and Ranganathan and Foster, 2003) have taken these three schedulers in two slots. The scheduler picks the server first, which already jobs have been processed and waiting i.e. Local Scheduler (*LS*). The data begin to be transferred immediately means Data Scheduler (*DS*) will start its work. It is possible that all jobs for that server are finished by the Execution Time (*ET*) that the transferring finishes so the time to enter service is merely the Data Transfer time (*DT*). It is also possible that some jobs are not finished when the data transfer completes so that the new job must wait until all previous jobs are completed. That time is just the system time (*ET*) of the current jobs at the server. Thus the total time that the new job must wait before service is the Queue Time (*QT*) i.e. system time (*ET*) for all jobs currently at that server. That is why above architectures are taking maximum of *QT* and *DT* during the transfer time calculation (Equation 1.1). According to these architectures the whole system is busy at a time, it means that *DS* will keep continue its work and meanwhile there will be jobs in the *ES*, and *ES* will execute jobs and at the same time, meanwhile there will be jobs in *LS*.

In order to compare the results of the new proposed technique and existing technique, we need to calculate the Queue Time (*QT*), Execution Time (*ET*), and Data Transfer Time (*DT*). By adding all these three parameters' values, we can get the *Total Completion Time (TCT)*. Suppose we need to calculate Transfer Time for 4 KB data by using 56Kbps bandwidth. The values can be compared (in terms of accuracy). According to the existing formula  $TT_{k,i} = \max\{QT_{(i)}, DT(f(k),i)\} + ET_{k,i}$  by (Nguyen and Lim, 2007; and Tang et al., 2006) the Total Time consumed in transferring is **5.125** sec, and according to the new proposed technique  $TCT_{ji} = QT_{(i)} + ET_{j,i} + DT(f(j))$ , Total Time consumed in transferring is **5.7101** sec. Even *TT* is less than *TCT*, but in *TT* one value either *QT* or *DT* has been ignored. Ignoring one important value effects the real and original results of the data transferring time calculations. For efficient data management, the accuracy in the transfer time calculation is more significant, which is the basic need for Cloud Computing service's provision.

## 1.6 OBJECTIVES OF RESEARCH

This research focuses on scheduling technique to support the calculation of the total completion time for transferring data from source to target.

The objectives of this research are:

- i. To propose and develop a new technique for the calculation of Total Completion Time (*TCT*) for data transferring process in cloud environment.
- ii. To analyse, test, and evaluate new technique mathematically.
- iii. Compare the performance of proposed techniques with existing models.

## 1.7 SCOPE OF RESEARCH

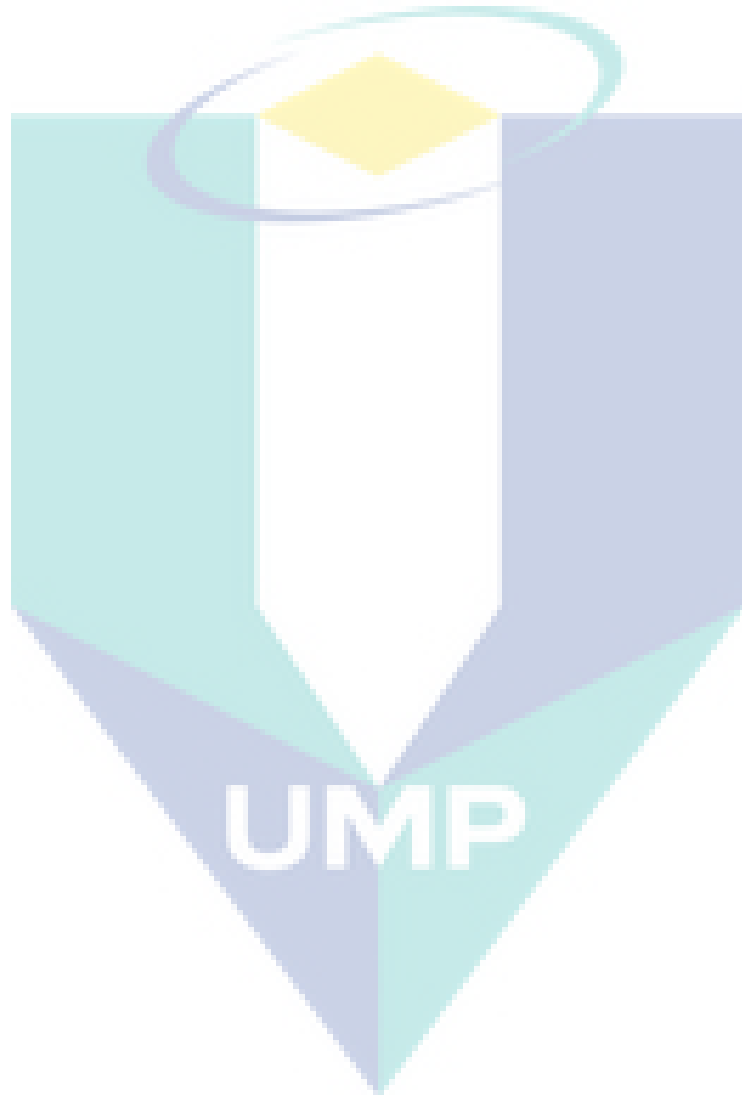
The scopes of this research are:

- i. All parameters have considered for use in Total Completion Time calculation procedure.
- ii. Finite population M/M/C/K/P queuing model has applied for evaluation.
- iii. Scheduling policy First-Come-First-Serve (FCFS) has used.

## 1.8 STRUCTURE OF THE THESIS

This thesis has been prepared to give details about the basic facts, calculations, arguments, and procedures in order to meet its objectives. Chapter 1 generally describes the background of data management, scheduling, problem statement, objectives, and scope of the research. Chapter 2 reviews the Cloud Computing, types of Cloud Computing, comparison with Grid Computing, different Cloud platforms, scheduling,

and various scheduling techniques in Cloud and Grid environment, queuing theory and different queuing models. Chapter 3 presents the new proposed scheduling architecture in cloud environment as well as framework, flowchart, and examples. Chapter 4 elaborates the implementation of results, discussion, and comparison with existing models. The conclusions of the present research are summarized and presented in Chapter 5 with suggestions and recommendations for future research.



## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 INTRODUCTION

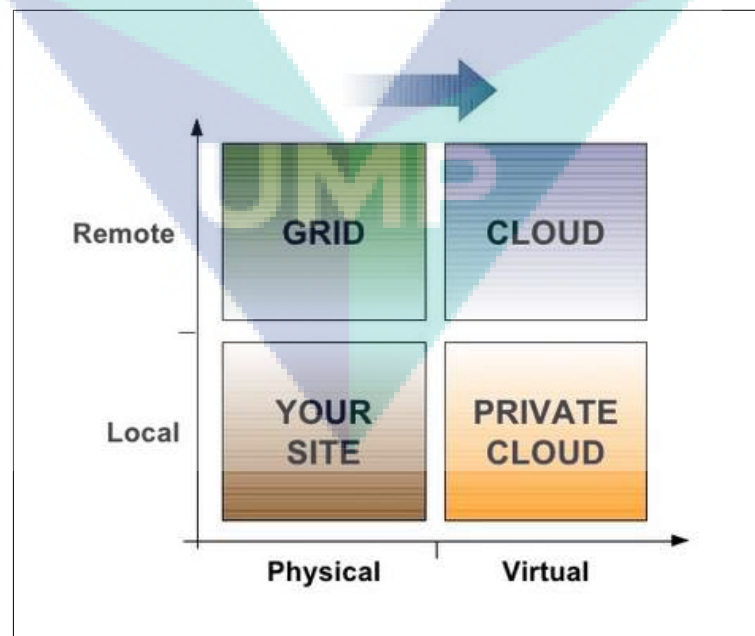
This chapter presents the background of Grid Computing and Cloud Computing, comparison of Grid with Cloud, Cloud types, and Cloud platforms. Cloud Computing efficiency is based on data management principles whereby scheduling is the most important category of the data management. This chapter briefly discusses scheduling principles and queuing models.

#### 2.2 GRID COMPUTING

Grid systems are a well-known technology that can provide a seemingly unique infrastructure from several resource providers, possibly heterogeneous (Luis et al., 2012). In other words, a Grid is a collection of computers, usually owned by multiple parties and in multiple locations, connected together such that users can share access to their combined power. Grid Computing (Grid) allow consumers to obtain computing on demand, analogous in form and utility of the electrical grid (Amazon, 2010). Grids and related application technologies are enabling scientists and engineers to build more and more complex applications for managing and processing large data sets, and for executing scientific experiments on distributed heterogeneous resources (Google, 2010). Cloud aim for the same dream of using computing as a utility (Armbrust et al., 2009). The fundamental vision and concepts are the same. The vision of a Global Grid has not

yet been realized but it might be fair to say that Cloud builds on the lessons learnt from building a Grid.

Typically, Grid users send their tasks to the Grid platform which will distribute them among the resources available. Activities such as resource location, execution scheduling, security handling, etc. are managed by the Grid. Grids can use Cloud as infrastructure providers so they can deploy or release resources in order to react to changes on demand, or to anticipate to variations on that demand if load prediction systems (like [Caron et al., 2010]) are available. This demand of resources will be induced by the amount (which depends on the triggering rate) and size of tasks sent to the Grid. Thus, Grids will be able to allocate only the infrastructure they required. Grid computing enables the sharing, selection, and aggregation by users of a wide variety of geographically distributed resources owned by different organizations and is well-suited for solving IT resource intensive problems in science, engineering and commerce. Grids are very large-scale virtualized and distributed computing systems. They cover multiple administrative domains and enable virtual organizations (Delic and Walker, 2008). Such organizations can share their resources collectively to create an even larger Grid.



**Figure 2.1:** Resource Provisioning Models

Besides, Grids can benefit from Cloud's flexibility as they will be able to run tasks with heterogeneous software requirements in the same host. Grids and clouds are much the same; both Grids and Clouds have adopted the concept of IT 'as-a-service', although Grids are more likely to offer free access to shared resources, while Clouds have a 'pay-as-you-go' approach. Figure 2.1 shows, the change in resource provision model from *physical* to *virtual* and from *local* to *remote* environment.

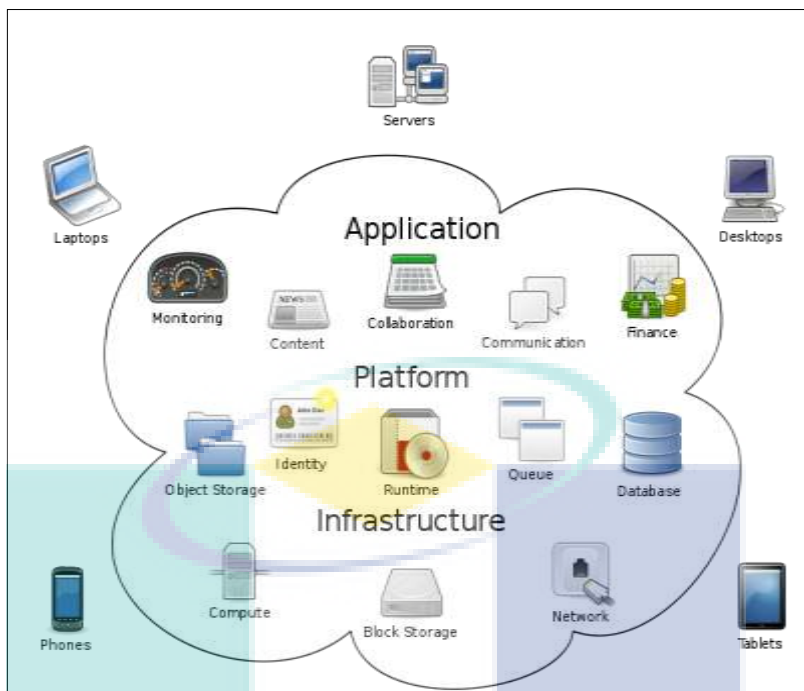
To get Cloud Computing to work, we need three things: thin clients (or clients with a thick-thin switch), Grid computing, and utility computing. Grid computing links disparate computers to form one large infrastructure, harnessing unused resources. Utility computing is paying for what you use on shared servers like you pay for a public utility (such as electricity, gas, telephone etc.).

### 2.3 CLOUD COMPUTING

Cloud Computing is hinting at a future in which we would not compute on local computers, but the user will use central facilities (compute, transfer, and storage utilities) which operates by a third-party. In fact, back in 1969, Computing inventor McCarthy predicted that "Computation may someday be organized as a public utility and let see how this speculates might occur" (Kleinrock, 2005), first time he publicly suggested in his talk in 1969 that computer time-sharing technology might lead to a near future in which computing power and even specific applications could be sold through the utility business model just like water, electricity, telephone, shops, and houses etc. for rent, under the rule of *Pay-As-You-Use*.

Forty two years after the prediction of McCarthy, in 2011 Rajkumar Buyya writes in his book that now "Cloud Computing is an utility computing model for on-demand delivery of computing power; consumer pay providers based on usage *Pay-As-You-Go*, similar to the way in which we currently obtain services from traditional public utility services such as water, electricity, gas, and telephony" by (Buyya et al., 2011).





**Figure 2.2:** Cloud Computing Simple Architecture

Source: IT ProPortal 2012

Cloud Computing has given the signal for the next new generation period which is a combination of computing and internet. As a result, the user will be able to access and store software, data, and content run by a third party enterprise in remote servers or by a client (Foster et al., 2008). The access device can be computers, phones, TVs, etc. Services are accessible through the internet anytime, anywhere, and from anywhere in the world as mentioned in Figure 2.2. Various companies or consumers had already started the usage of Cloud applications such as iTunes, Hotmail, Yahoo, Gmail, Google Docs, Google Earth, Online Operating System, Facebook, and Flickr etc. Hence Cloud Computing is a Web-based process and service, whereby shared resources, information, and software's are provided to the consumer through various devices on demand over the Internet.

In the contextual of Cloud Computing, the term Grid was invented in the mid-1990s to describe and define technologies that would allow the users to obtain and use computing power on demand when they need. By standardizing the protocols used to

request computing power, (Foster et al., 2008) further explains that we could encourage the creation of a Computing Grid, and now Cloud Computing is comparable to the electric power Grid system in terms of delivering utility in the form of services.

A large-scale distributed computing model that is driven by economies of scale, in which an abstracted virtualization, managed by computing power, storage, dynamically-scalable hypervisor, platforms, and services are delivered on demand to external jobs over the internet (Jeremy et al., 2011), as Figure 1.1 has described some benefits, features, modes type, and locality of the Cloud system. Below section discusses some benefits of Cloud Computing.

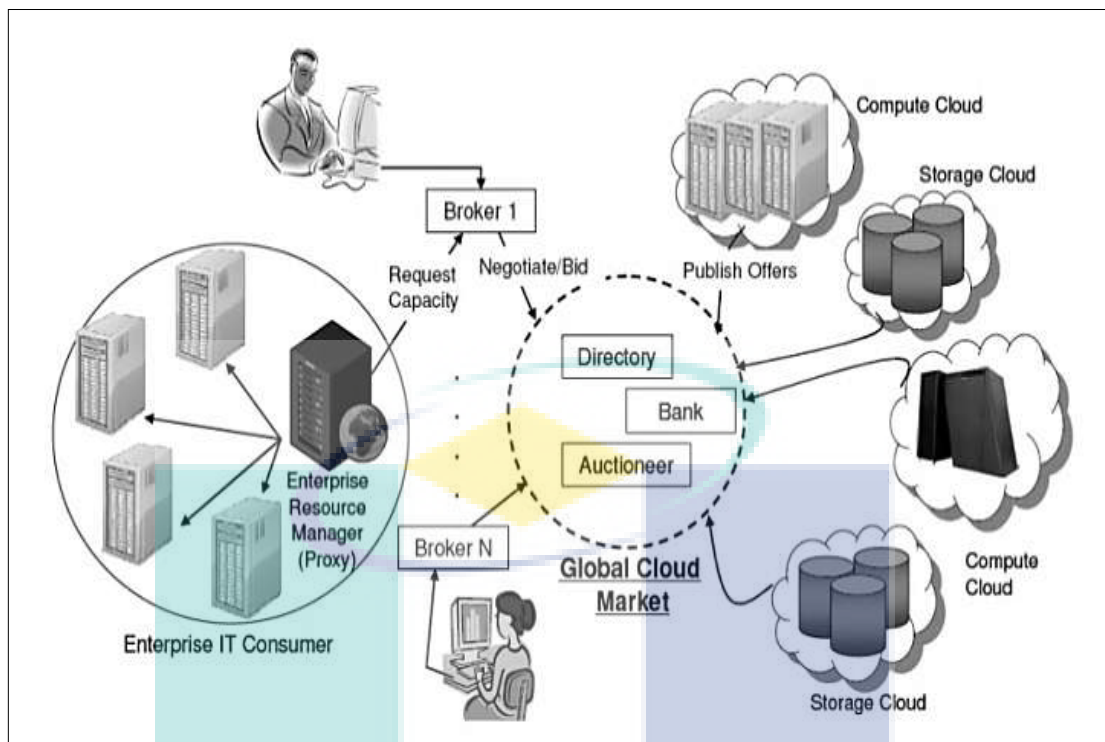
### 2.3.1 Major Benefits

- *Lower Costs:* Basically, Cloud Computing is the computing resource and delivery which shows a better efficiency and utilization of the whole shared infrastructure.
- *No Ex-Capital:* Public Cloud or private Cloud, both deliver a better cash flow by excluding the capital expense associated with the structure of the server infrastructure.
- *Faster Deployment:* Servers are available to take and leave anytime, as per requirement, just in a matter of minutes. The time to deploy new application drops with Cloud Computing.
- *Scale as Needed:* During the growing or declining in applications, for *just enough* scale, the user can add or remove storage, RAM, and computing capacity as needed.
- *Lower Maintenance Costs:* With less physical resources in outsource environment, there is less hardware to power and maintain. There is no need to keep specialists in matters of storage, server, network, and virtualization on a full-time basis.
- *Resiliency and Redundancy:* Especially in private, the Cloud deployment user can get automatic fail over between hardware platforms and disaster-recovery services, to manage and bring up server to set in a separate data centre as a primary data hub.

Cloud Computing is emergent based on years of achievement on Grid Computing, Virtualization, Utility Computing, Web Computing, and related technologies. Cloud Computing provides both platforms and applications on-demand through the internet or intranet as discussed by (Foster et al., 2008; Dean and Ghemawat, 2004). Some examples of emerging Cloud Computing platforms are Amazon EC2 (Amazon EC2, 2011), IBM blue Cloud (IBM, 2011), Google App Engine (Google App Engine, 2011), and Microsoft Azure. The Cloud allows sharing, aggregation, and allocation of software, storage, and computational network resources on-demand. Some of the key benefits of Cloud Computing include hiding and abstraction of complexity, virtualized resources, and efficient use of distributed resources had discussed by (Foster et al., 2009).

So *Cloud Computing* is a tremendous revolution in Grid Computing, which has the same vision. Both focus on reducing the cost of computing and increasing reliability and flexibility by transforming computers from something that we buy and operate ourselves to something that is operated by a third party (Jeremy et al., 2011). Now things are different than they were ten years ago. With the passage of time, users have a new need to analyse huge data, thus motivating greatly increased demand for computing. Enterprises have spent multiple billions of dollars on various services, e.g., Amazon, Google, and Microsoft to create real commercial large-scale systems containing thousands of thousands of computers.

There are so many comprehensive definitions of Cloud Computing by various researchers. The definition created by (Foster et al., 2008) is *A large-scale distributed computing paradigm that is driven by economies of scale*, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external jobs over the internet (Jeremy et al., 2011), as shown in Figure 2.3.



**Figure 2.3:** Global Cloud exchange Infrastructure

Source: Buyya et al., 2008

### 2.3.2 Cloud comparison with Cluster and Grid

A number of computing researchers and practitioners have attempted to define Clusters, Grids, and Clouds (Jeremy et al., 2011) in various ways. There are some definition and comparisons of Cluster and Grid with Cloud Computing, have given in Table 2.1.

Cluster define by (Buyya, 1999) as ‘A *Cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource*’.

“A *Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed ‘autonomous’ resources*

*dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements” defined by (Buyya et al., 2009).*

Based on the observation of what Clouds promise to be, proposed the following definition by (Buyya et al., 2009): “A *Cloud* is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and the consumers”.

**Table 2.1:** Characteristics Comparison of Cluster, Grid, and Cloud Computing

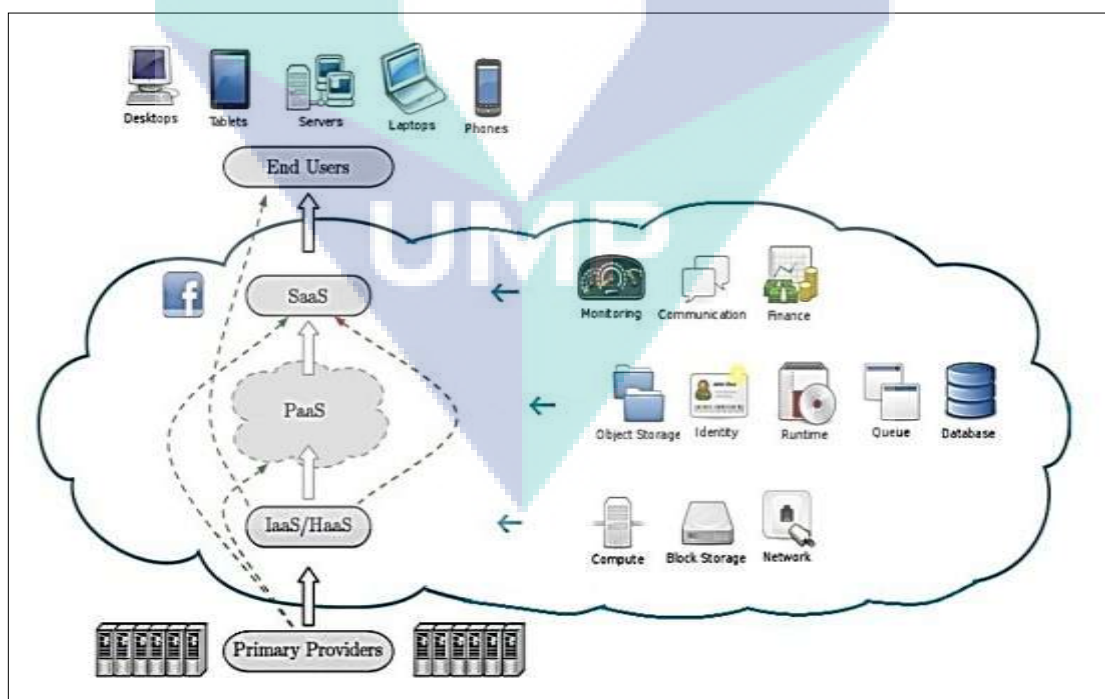
Characteristics	Computing Environment / Systems		
	Clusters	Grids	Clouds
Population	Commodity computers	High-End computers (Servers, Clusters)	Both and also web-based storage
Scalability	100s	1000s	100s to 1000s
Service Negotiation	Limited	Yes, SLA based	Yes, SLA based
User Management	Centralized	Decentralised and also virtual organization based	Centralised or can be delegated to third party
Resource Management	Centralized	Distributed	Centralized/Distributed
Capacity	Stable and Guaranteed	Varies, but high	Demand-based Provisioned
Pricing of Services	Limited, not open market	Dominated by public good or privately assigned	Utility pricing, discounted for larger jobs
Node Operating System (OS)	Standard OS (Windows, Linux)	Standard OS (Dominated by Unix)	A hypervisor (VM) which can run multiple OSs
Failure Management (Self-Healing)	Limited (Often failed task/application are restarted)	Limited (Often failed task/application are restarted)	Strong support for failover and content replication. VMs can be easily migrated from one node to other.
Ownership	Single	Multiple	Single
Interconnection	Dedicated, high-	Mostly Internet	Dedicated, high-end

Network/Speed	end with low latency and high bandwidth	with high latency and low bandwidth	with low latency and high bandwidth
Security/Privacy	Need Traditional login; Medium level of privacy- depends on user rights	Public/private key base authentication and mapping a user to an account. Limited support for privacy.	Each user/application is provided with a virtual machine. High security/privacy is guaranteed.
Discovery	Membership service	Centralised indexing and decentralised info services	Membership services
Allocation/Scheduling	Centralized	Decentralized	Both Centralized/Decentralised
Standards/Inter-Operability	Virtual Interface Architecture (VIA)-based	Some Open Grid Forum Standards	Web Service (SOAP and REST protocols )
System Image Single	Yes	No	Yes, but optional
Interworking	Multi-clustering within an organization	Limited adaptation, but being explored through research efforts such as Gridbus, InterGrid	High potential, third party solution providers can loosely tie together services of different Clouds
Drivers for Application	Science, business, enterprise, computing, data centres	Collaborative scientific and high output computing applications	Dynamically provisioned legacy and web application, content delivery
Building Potential 3rd-Party or Value-Added Solution	Limited due to rigid architecture	Limited due to strong orientation for scientific Computing	High potential can create new services by dynamically providing compute, strong and application services and offer as their own isolated or composite Cloud services to users.

Source: Buyya (2009)

### 2.3.3 Types of Clouds

Cloud Computing generally can provide its services in three styles, Infrastructure-as-a-Service (*IaaS*), Platform-as-a-Service (*PaaS*), and Software-as-a-Service (*SaaS*). *SaaS* means the presented service to a user is the utility running on the infrastructure of Cloud Computing. It is easy accessible by thin user interfaces such as a simple browser. *PaaS* refers to the organisation of services produced by the development of language and tool, for instance, python, Java, and .net which are delivered by the service presenters/providers to the infrastructure of Cloud. *IaaS* refers to the utilities/services presented/provided to the consumers by leasing the power of processing, data transpiring capabilities, network, storage power, and other elementary computing resources, with which the consumers can deploy and run any application, including operating systems and other services. In all these services/utilities, there is no need for the users to manage or control the operating system, cloud's infrastructure, server, network, storage, and even the application's functions. For more explanation Figure 2.4 describes the whole categorization of services in Cloud.



**Figure 2.4:** Cloud Types

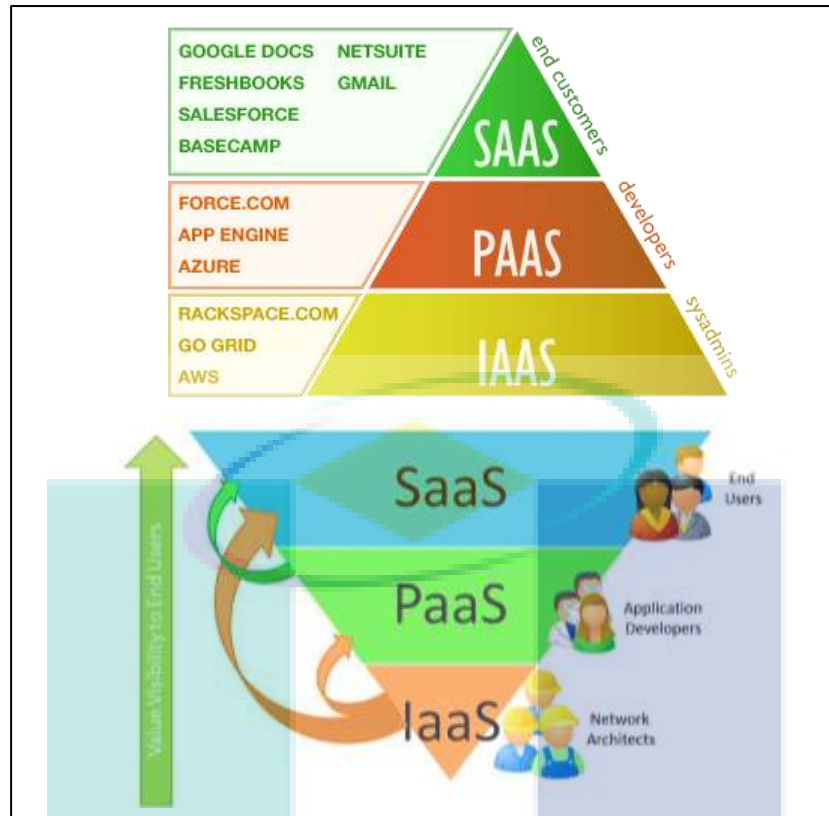
### 2.3.3.1 Infrastructure-as-a-Service (IaaS)

Computing is being transformed to a model consisting of various utilities that are commoditized and delivered in a way like a traditional utility such as water, gas, electricity, and telephony. Several computing models have offered successful delivery of all utilities of computing, data storage, and data transferring. Computing vision includes Cluster Computing and Grid Computing, then more recently Cloud Computing.

The computing world is rapidly changing in the direction of developing software for millions of users to consume utilities as a service, rather than to run on their specific, reserved, and on individual computers (Buyya et al., 2008; and Google App Engine, 2011). In such a paradigm, clients can access services based on their requirements without reference to where the location for service is or how they are delivered. Internet Providers (*IPs*) manage a massive set of computing resources, such as storage resources and processing power capacity. By using virtualization, the providers have capabilities to divide, allocate, dynamically resize, and reshape these resources to build ad-hoc systems according to the needs and request of the user by jobs as well as Service Providers (*SPs*). They make set-ups for software stacks which can run their services/utilities, by using their Infrastructure-as-a-Service (*IaaS*) scenario (Peng and Zhang, 2009; Rimal et al., 2009; and Google App Engine, 2011).

Infrastructure-as-a-Service is also regarded as resource Clouds, which provides managed and scalable Resources-as-a-Service (*RaaS*) to the user, in other words, they basically provide enhanced virtualisation capabilities. Accordingly, different resources may be provided via a service interface. Data and storage clouds will deal with reliable access to data of potentially dynamic size, and weigh resource usage with access requirements and/or quality definition. Some examples of the Infrastructure-as-a-Service (*IaaS*) are GO GRID, RACKSPACE.COM, Amazon Elastic Compute Cloud (*EC2*), Zimory, Elastic Hosts, and Amazon Web Service (*AWS*), as stated in Figure 2.5.





**Figure 2.5:** Cloud Services Structure

Source: IBM Developer Works 2009

### 2.3.3.2 Platform-as-a-Service (PaaS)

Platform-as-a-Service (*PaaS*) provides computational resources as utilities through a platform in which applications and services can be developed and hosted. (Peng and Zhang, 2009; Rimal et al., 2009; Buyya et al., 2008; and Google web-resource, 2011) have discussed platforms in detail. *PaaS* typically makes use of dedicated Application Programming Interfaces (*APIs*) to control the performance of a server hosting engine which executes and replicates the execution according to user requests (e.g. access rate). As each provider exposes his/her own *API* according to the respective key capabilities, applications developed for one specific Cloud provider cannot be moved to another Cloud host, there are, however, attempts to extend generic programming models with Cloud capabilities (such as Microsoft Azure). Few examples

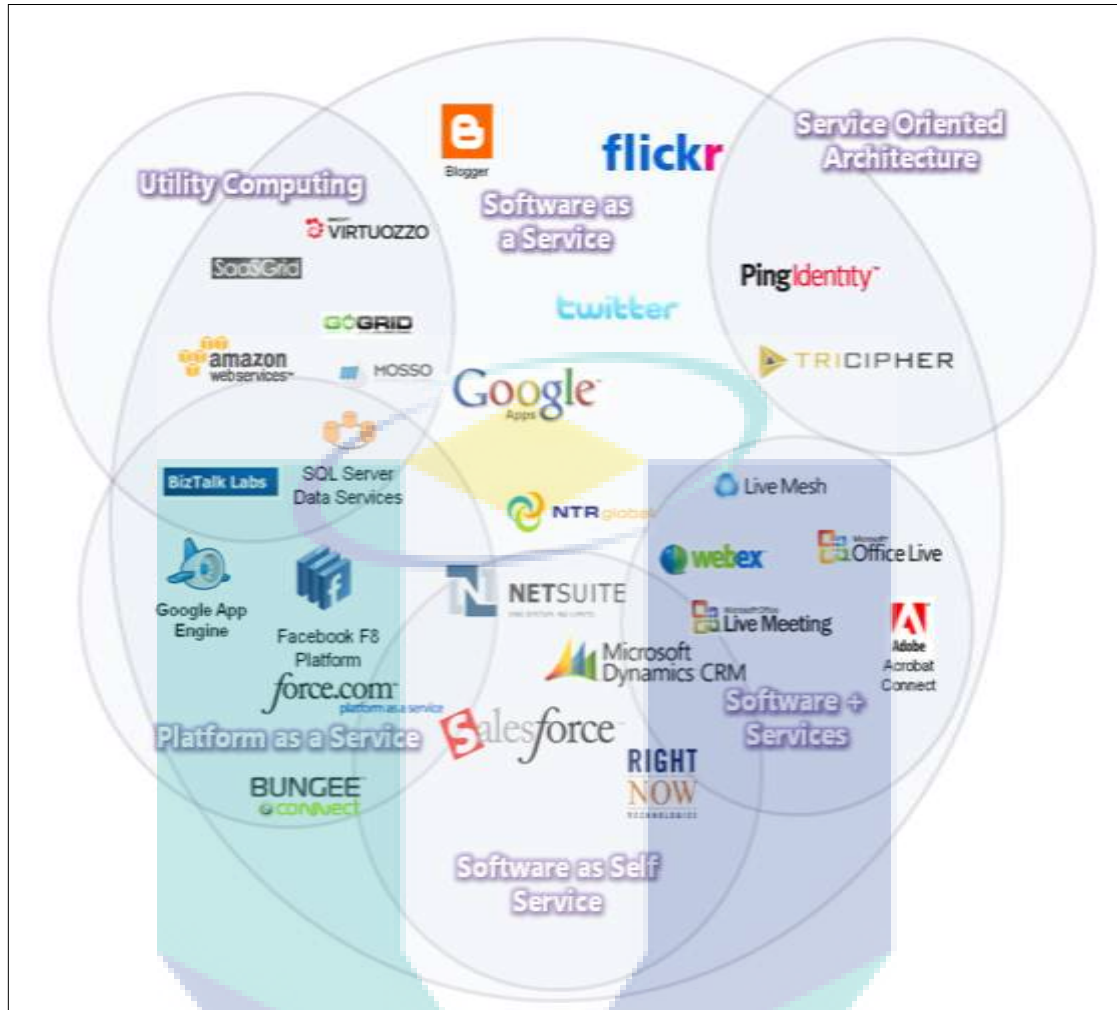
of Platform-as-a-Service are Force.com, Google App-Engine and Windows Azure, as shown in Figure 2.5.

Cloud Computing provides access to computational resources, i.e. CPUs. So far, such low-level utilities cannot really be exploited on their own, so that they are typically exposed as part of a *Virtualized Environment*, i.e. Virtual Machine (hypervisors). Compute Cloud Providers therefore typically offer the capability to provide computing resources (i.e. raw access to resources unlike *PaaS* that offer full software stacks to develop and build applications), typically virtualised, in which to execute Cloudified services and applications.

#### 2.3.3.2.1 PaaS Features

In the world (Global village), there are different Cloud Computing platforms, whereas each platform has its own benefits and features (Rimal et al., 2009). For better understanding, after analysing and comparing these platforms, it will be easy for the user to select the best and suitable implementation aspects for his/her business. Table 2.2 compares the capabilities, advantages, and opportunity of some platforms, e.g. Elastic Compute Cloud (EC2), Azure, App-Engine, Sun Grid, and Aneka.

*PaaS* contributes facility in the management of applications without any cost and difficulties of buying and managing the underlying software as well as hardware. *PaaS* has the ability to present hosting capabilities, offer all services required to give hands-in completion of life cycle of structure and deliver web applications/software, whereas these services are entirely available via the internet (Google web resource, 2011). Elastic Compute Cloud (*EC2*) from Amazon, *Apps Engine* from Google and *Azure* from Microsoft are some examples of platforms, as categorized in Figure 2.6.



**Figure 2.6:** Cloud Computing Taxonomy Map

Source: Matias 2008

*PaaS* presents facilities and services for application design, development, testing, deployment, and hosting as well as applications (utilities) such as team partnership, web services/utilities integration, integration of database, security of data, scalability in software and hardware, marshalling, storage, state management, persistence, application versioning, application arrangement/instrumentation, and developer community facilities (Peng and Zhang, 2009). These utilities/services are provisioned as a combined solution over the web. Table 2.2 describes some comparison of various *PaaS*.

**Table 2.2:** Comparison of various platforms

Property	Different Platforms				
	Amazon Elastic Compute Cloud (EC2)	Google App Engine	Microsoft Azure	Sun Network.com (Sun Grid)	GRIDS Lab Aneka
Focus	Infrastructure	Platform	Platform	Infrastructure	Enterprise Clouds
Service Type	Compute, Storage (Amazon S3)	Web-Application	Web and non-web application	Compute	Compute
Virtualization	OS level running on a Xen hypervisor	Application container	OS level through fabric controller	Job management system (Sun Grid Engine)	Resource manager and scheduler
Dynamic negotiation of QoS	None	None	None	None	SLA-based resources reservation
Web APIs	Yes	Yes	Yes	Yes	Yes
User Access interface	Amazon EC2 command-line tools	Web-based administration	Microsoft windows azure portal	scripts, Sun Grid web portal	Work-bench, web-based portal
Value-added service providers	Yes	No	Yes	Yes	No
Programming frame-work	Amazon Machine Images (AMI)	Python	Microsoft.NET	Solaris OS. Java, C, C++, FORTRAN	APIs supported models in c#.Net

Source: Buyya (2009)

User necessities for Cloud services are varied, service providers need to ensure that they can be flexible in their service delivery while keeping the users isolated from the original infrastructure. Recent improvements in microprocessor technology and software have led to the growing ability of commodity hardware to run utilities (applications) within Virtual Machines (VMs) efficiently. VMs allow both the inaccessibility of applications from the underlying hardware and other VMs and the customization of the platform to suit the needs of the end-user.

### 2.3.3.3 Open Source Cloud

The starring role of open source Cloud Computing is to build some mechanism around digital identity management and outlines some technological building blocks which are needed for controllable confidence and identity verification. Eucalyptus, Open Nebula, and Nimbus are technically sound and popular. Current Cloud has a focus on the issue of interoperability, which is essential for the enter-prisesed Cloud system. Table 2.3 presents brief comparison of these open-source platforms.

**Table 2.3:** Comparison of open-source Cloud platforms

Feature	Eucalyptus	OpenNebula	Nimbus
Computing Architecture	<ul style="list-style-type: none"> <li>-Ability to configure multiple clusters, each with private internal network addresses, into a single Cloud.</li> <li>-Private Cloud</li> </ul>	<ul style="list-style-type: none"> <li>-Cluster into an IaaS Cloud</li> <li>-Focused on the efficient, dynamic, and scalable management of VMs within data centres (private Cloud) involving a large amount of virtual and physical servers</li> <li>-Based on Haizea scheduling</li> </ul>	<ul style="list-style-type: none"> <li>-Science Cloud</li> <li>-Client-Side Cloud-Computing interface to Globus-enabled TeraPort cluster</li> <li>-Nimbus Context Broker that combines several deployed virtual machines into “turnkey” virtual clusters</li> <li>- Heterogeneous clusters of auto-configuring VMs with one command</li> </ul>
Virtualization Management	-Xen hypervisor	-Xen KVM and on-demand access to Amazon EC2	-Xen Virtualization
Service	IaaS	IaaS	IaaS
Load Balancing	-Simple load-balancing Cloud controller	-Nginx Server conFIGured as load balancer, used round-robin or weighted selection mechanism	-Launches self-configuring virtual cluster i.e. the context broker
Fault	-Separate cluster	-The daemon can be	-Checking worker nodes

Tolerance	within the Eucalyptus Cloud reduce the chance of correlated failure	restarted and all the running VMs recovered -Persistent database backend to store host and VM information	periodically and recovery
Interoperability	-Multiple Cloud Computing interfaces using the same “back-end” infrastructure	-Interoperable between intra Cloud services	-Standards : “rough consensus and working code”
storage	-Walrus (the front end for the storage subsystem)	-Database, persistent storage for ONE data structures -SQLite3 backend is the core component of the Open Nebula internal data structures	-Grid FTP and SCP
Security	-WS-security for authentication, Cloud controller generates the public/private key	-FIRDeelman, E.II, Virtual Private Network Tunnel	-PKI credential required -Works with Grid proxies VOMS, Shibboleth (via Grid Ship), custom PDPs
Programming Framework	-Hibernate, Axis2 and Axis2c, Java	-Java, Ruby	Python, Java

Source: Buyya (2009)

### 2.2.3.3.1 Open Source Cloud Platforms Comparison

There are different types of Cloud platforms each one has its own characteristics and advantages. After analysis and comparison, for ease of understanding, Table 2.4 discusses in detail its implementation aspects.

**Table 2.4:** Comparing platforms with implementation aspects

	<b>Eucalyptus</b>	<b>Nimbus</b>	<b>Open Nebula</b>
Cloud Character	Public	Public	Private
Scalability	Scalable	Scalable	Dynamical, Scalable
Cloud Form	IaaS	IaaS	IaaS
Compatibility	Support EC2, S3	Support EC2	Open, Multi-Platform
Deployment	Dynamic Deployment	Dynamic Deployment	Dynamic Deployment
Deployment Manner	Command line	Command line	Command line
Transplantability	Common	Common	Common
VM Support	VMWare, Xen, KVM	Xen	Xen, VMWare
Web Interface	Web Service	EC2 WSDL, WSRF	Libvirt, EC2, OCCI API
Reliability	-	-	Rollback host and VM
OS Support	Linux	Linux	Linux
Development Language	Java	Java, Python	Java

Source: Buyya (2009)

#### 2.3.3.4 Software-as-a-Service (SaaS)

Software-as-a-Service (*SaaS*) refers to a Cloud service or application, which offers the implementations of specific business functions and processes. *SaaS* is being provided with particular Cloud capabilities, i.e., they provide applications/services/utilities by using a Cloud infrastructure or platform, rather than providing Cloud features (Buyya et al., 2008). Often, this kind of standard application software is functionally presented within a Cloud. *SaaS* gives quick access to value the standard business process. *SaaS* is highly suitable for those enter-prises areas where the processes need to be standardized (Rimal et al., 2009). Almost, however, these

processes are not equivalent because they are actually sub-standard. If businessmen agree to accept standardized business procedures and the pre-built *SaaS* packages which can mechanize those standardized processes, they would meaningfully improve and expand their enterprise/business. In these situations where the data and information are too sensitive to believe offsite with a *SaaS* seller, then for the safety of data, the *SaaS* application frequently can be hosted in the business/enterprise's data centre (Buyya et al., 2008; and Rimal et al., 2009). Figure 2.7 presents general taxonomy of cloud including *SaaS* and its services.

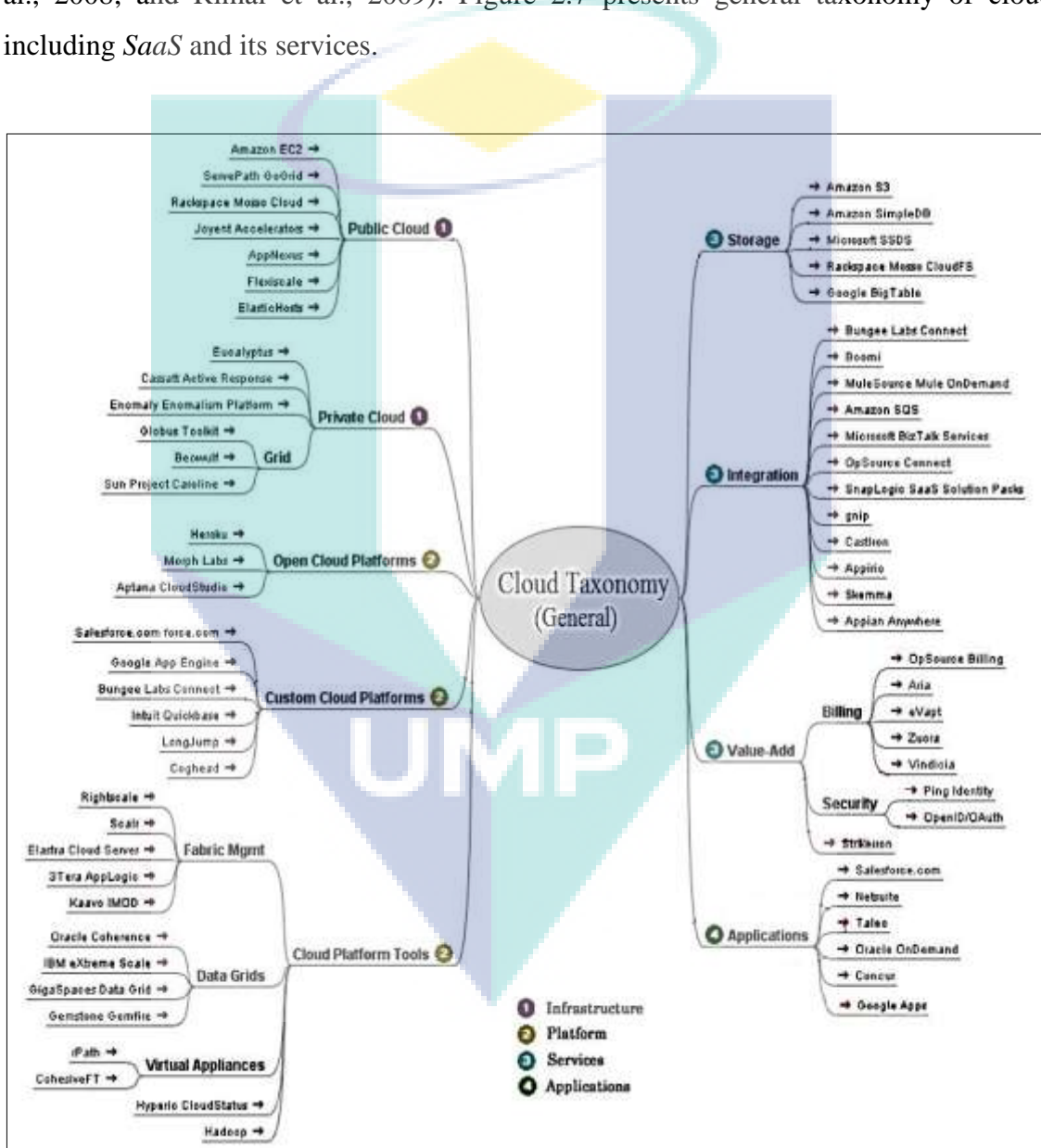


Figure 2.7: Cloud Computing Taxonomy

Source: Kaskade 2009



Figure 2.7 presents brief Taxonomy of Cloud Computing. *SaaS* is one of the main and important parts of Cloud Computing. Enterprises, based on their requirements and desired security needs, should consider internal, public, and private options of Cloud Computing. *SaaS* has the potential to provide the users various varieties hosted in Cloud Computing. This is an alternative to locally run applications/utilities; an example of this typical office application/software is word processors (Rimal et al., 2009; and Google, 2011). Some examples of the *SaaS* are *Google Docs* by Google, *CRM* by Sales force, *Office Live* and *Window Live* from Microsoft.

### 2.3.3.5 Data-as-a-Service (DaaS)

Data-as-a-Service (*DaaS*) is an information provision and distribution model in which data files (including text, images, sounds, and videos) are made available to customers over a network, typically the Internet (SearchCloud, 2013). *DaaS* offers convenient and cost-effective solutions for customer and client oriented enterprises. Few examples of *DaaS* providers are:

- *Fidelitone*: A supply-chain and logistics management company, employed ARI's DataStream DaaS solution to deploy parts catalogues into the customer channel.
- *Urban Mapping*: A geography data service which provides data for customers to embed into their own websites and applications.
- *Xignite*: A company that makes financial data available to customers.
- *Hoover's*: Provides customers with business data on various organizations.

*DaaS* is emerging as underlying technologies that support Web services and SOA (*Service-Oriented Architecture*) mature. High-speed Internet service has become increasingly available to support user access from more areas around the world, making *DaaS* an attractive option to more people and organizations. The evolution of *SOA* has greatly reduced the relevance of the particular platform on which data resides.

Few benefits of *DaaS* are listed below.

- Ability to move data easily from one platform to another.
- Avoidance of the confusion and conflict that can occur when multiple copies and same data exist in different locations.
- Reducing overall cost of data maintenance and delivery.
- Preservation of data integrity by implementing access control measures such as strong passwords and encryption.
- Ease of administration.
- Ease of collaboration.
- Compatibility among diverse platforms.
- Global accessibility.
- Automatic updates.

*DaaS* is expected to facilitate new and more effective ways of distributing and processing data. Information management specialists believe that as more companies figure out which data assets they can rent for competitive advantage, the *DaaS* market will continue to expand. *DaaS* is closely related to Storage-as-a-Service (abbreviated SaaS) and Software-as-a-Service (also abbreviated SaaS) and may be integrated with one or both of these provision models. As is the case with these and other Cloud Computing technologies, *DaaS* adoption may be hampered by concerns about security, privacy, and proprietary issues. *DaaS* is totally concern with data transferring, hence this study and new proposed technique totally related with *DaaS*. Active provision of data depends on the correct time calculation for data transferring process.

#### **2.2.4 Cloud Prices**

Several Cloud Computing and Conventional Computing data-centers are being built in seemingly surprising locations, such as Texas, San Antonio (Microsoft, US National Security Agency) and Quincy, Washington (Google, Yahoo, Microsoft). *DaaS* (Data-as-a-Service) allows for, but does not require, the separation of data cost and

usage from software or platform cost and usage. Hundreds of *DaaS* vendors, with various pricing models, exist worldwide. Pricing can be volume-based (a fixed cost per megabyte of data in the entire repository) or format-based (a fixed price per text file, another fixed price per image file, etc.).

The motivation and inspiration behind picking these locations is that the expenses for electric power, labour, property purchasing consumption, cooling cost, and tax charges are geographically variable and above all. Electricity costs and cooling charges can account for a third of the costs of a data centre. Although prices are fluctuating, we match present Cloud services rates. As a common and very beneficial example is Elastic Compute Cloud (EC2) presented by Amazon Web Services (AWS) which sells 1.0GHz x86 ISA ‘slices’ for \$0.10 per hour, and a new ‘slices’ or instance further can get just the cost of 2-5 minutes. Amazon’s Scalable Storage Service (S3) rates are \$0.12 to \$0.15 per GB/month, with additional bandwidth cost of \$0.10 to \$0.15 per GB to transfer data IN and OUT over the internet. Table 2.5 presents more comparison.

**Table 2.5:** Cost comparison for some platforms

<b>Platforms</b>	<b>Storage \$/GB/month</b>	<b>Transferring \$/GB</b>	<b>Computing \$/GHz/h</b>
Amazon EC2	0.055	0.10	0.100
Google App Engine	0.150	0.11	0.100
Microsoft Azure	0.150	0.13	0.120

## 2.4 DATA MANAGEMENT IN CLOUD COMPUTING

The concept of data management is relatively simple. Cloud services need access to high-quality, relevant data, provided in a timely and cost-effective manner. Access to accurate data is necessary for effective investment decisions, trade execution,

securities pricing, risk management, regulatory compliance, and portfolio valuation and measurement. Delivering on this concept, however, is challenging, particularly in a dynamic environment marked by significant regulatory changes.

Data management is an important aspect particularly in storing cloud services, where data is flexibly distributed across multiple resources (Shi et al., 2010). Cloud services are just data transportation. Implicitly, data consistency needs to be maintained over a wide distribution of replicated data sources. At the same time, the system always needs to be aware of the data location during replication. For taking decision of *when* and *where* to execute data, Cloud providers need efficient scheduling models, where it is the key factor in data management, to deliver services with effective scalability and reliability (Deelman and Ann, 2008).

Cloud services need to consider key elements such as data architecture, metadata security and storage. When these elements are properly organized into an effective data management initiative, firms can realize significant benefits including lower operating costs, better risk management, and fewer and less costly errors. Scheduling is the base element and most important part of the data management.

#### **2.4.1 SCHEDULING**

With virtualized manner platforms, computing power, storage, and different services are delivered on demand to external jobs over the internet (Deelman and Ann, 2008). *Right data in the right place at the right time* is efficient data management, which is the basic need of Cloud Computing services (Shi et al., 2010). Scheduling is the main factor of data management, and a queue phenomenon is the most important portion of scheduling. Table 2.6 compares some well-known scheduling algorithms with various aspects.

**Table 2.6:** Scheduling Algorithms Comparison

Scheduling Algorithm	CPU Overhead	Turnaround Time	Throughput	Response Time
First-In-First-Out	Low	High	Low	Low
Shortest-Job-First	Medium	Medium	High	Medium
Priority based scheduling	Medium	High	Low	High
Round-Robin scheduling	High	Medium	Medium	High
Multilevel Queue scheduling	High	Medium	High	Medium

#### 2.4.1.1 Scheduling Architectures

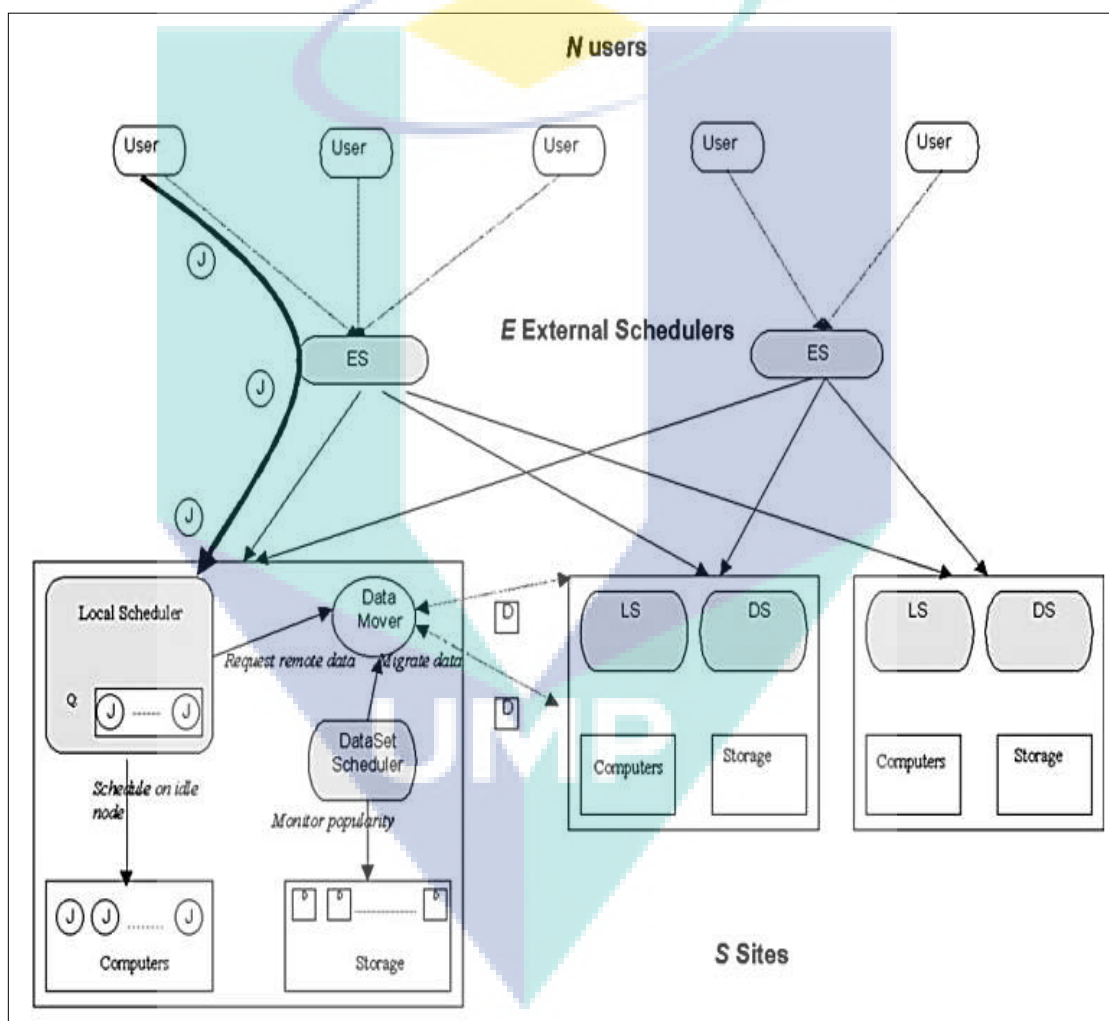
Ranganathan and Foster (2003) defines a scheduling architecture that facilitates efficient data management, allows priority to local policies, is decentralized with no single point of failure and employs online job allocation techniques. The scheduling logic of the architecture is encapsulated in three distinct modules Figure 2.8.

*External Scheduler (ES):* Each user in the system is associated with an External Scheduler and submits jobs to that External Scheduler. We can typically imagine one ES per site but the framework does not enforce this. The ES then decides the remote site to which to send the job to depending on some scheduling algorithm. It may use external information such as load at a remote site or the location of a dataset, as input to its decisions.

*Local Scheduler (LS):* Once a job is assigned to run at a particular site (and sent to an incoming job queue of that site) it is managed by the Local Scheduler at that site. The LS decides how to schedule all jobs allocated to it, on its local resources. It could,

for example, decide to give priority to certain kinds of jobs, or it could refuse to run jobs submitted by a certain user.

*Dataset Scheduler (DS):* The *DS* at each site keeps track of the popularity of each data set locally available. It then replicates popular datasets to remote sites depending on some algorithm. The *DS* may use external information such as whether the data already exists at a site and load at a remote site to guide its decisions.

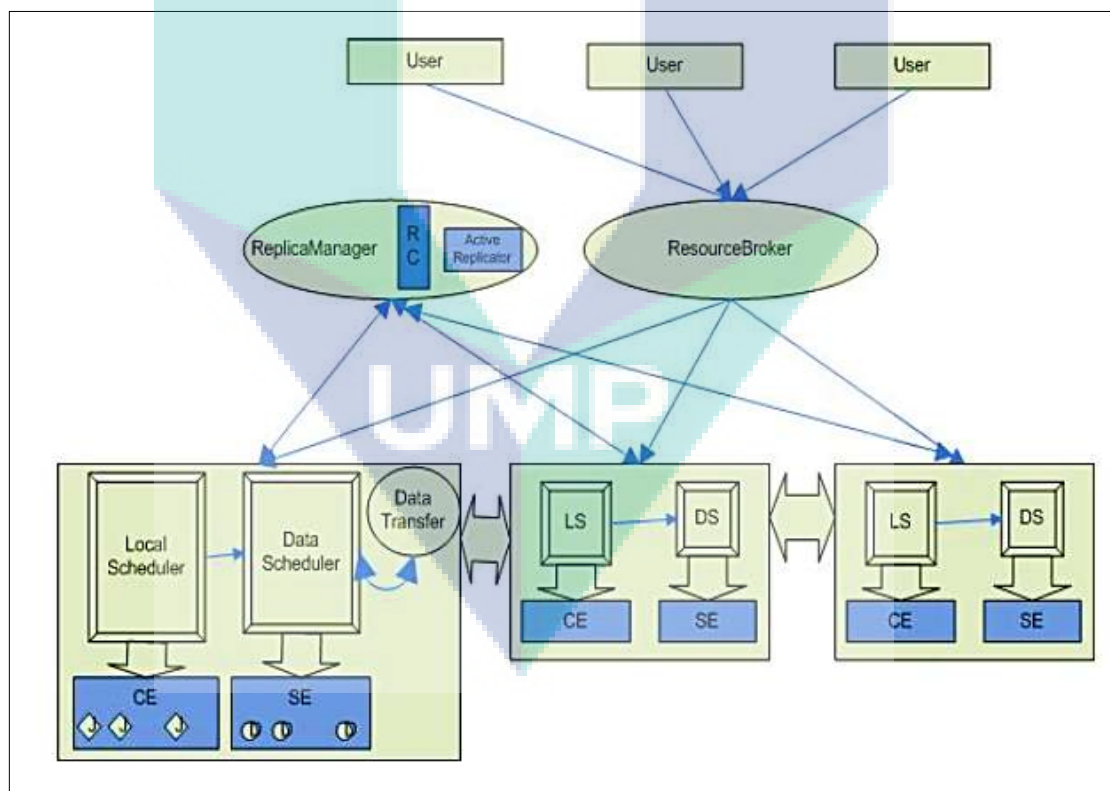


**Figure 2.8:** Job Scheduling Architecture A

Source: Ranganathan and Foster, 2003

Then (Nguyen and Lim, 2007) extends the work of (Ranganathan and Foster, 2003) and combines scheduling and replication as shown in Figure 2.9. However, they add into that model a centralized Active Replicator as part of Replica Manager and also modify the role of Data Scheduler. They used some terms i.e. LS/DS/CE/SE/RB: Local Scheduler/ Data Scheduler/ Computer Element/ Storage Element/ Resource Broker.

*Replica Catalogue (RC)*: Stores the list of all replicas on the grid. For every predefined interval, the replicator will collect the replica information and data usage information over the Grid. Then it decides whether to replicate a data file. When a job is assigned to *LS*, the *DS* will responsible for all the data requests by the *LS*. This data request is generated as soon as job is scheduled into *LS* queue. The objective of this *DS* is to obtain to the local site as much data required by a job as possible before that job is executed.



**Figure 2.9:** Jobs Scheduling Architecture B

Source: Nguyen and Lim, 2007

The basic system running procedure scheduler described by (Liang and Shi, 2010) as follows and described in Figure 2.10.

*Task submission:* A task is submitted from the submission site to the G-Scheduler; all tasks will stay in the G-Queue.

*Global scheduling:* The G-Scheduler schedules the task with First-Come-First-Serve (FCFS) policy from the G-Queue, and then selects the running sites based on the inter-site trust information (or sequential selection if without a reputation mechanism), and dispatches the job to the selected sites meeting the scheduling requirements. At the same time, the G-Scheduler changes the stock of currency between the submission site and selected running sites based on the reputation information and resource request. The selected running sites will put the job assignments from the G-Scheduler into the Local scheduler's Queue.

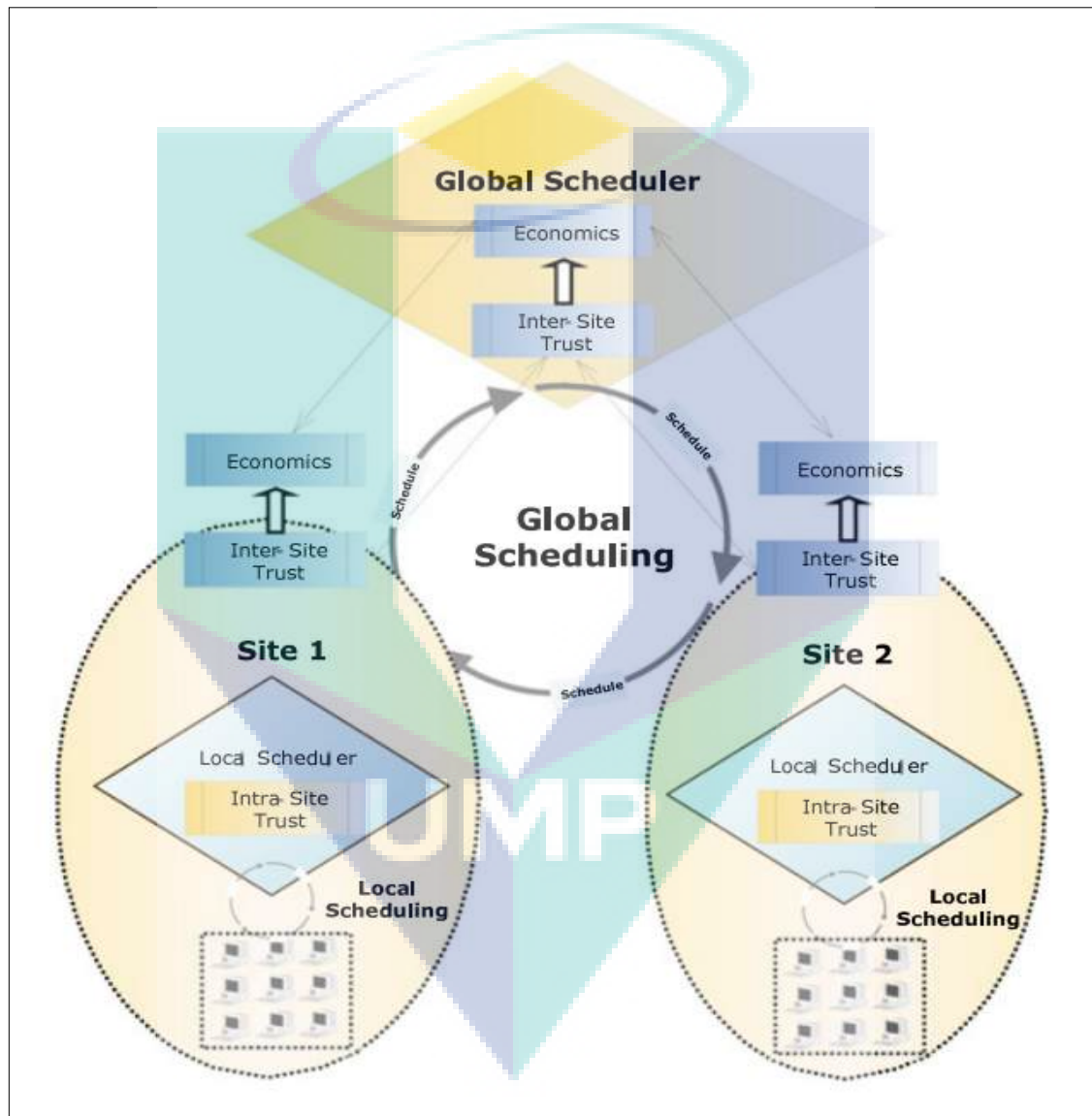
*Local scheduling:* The L-Scheduler in the running site also schedules jobs from the L-Queue with FCFS, makes node selection based on the intra-site trust information (or sequential selection if without a reputation mechanism), and dispatches the job to the selected running nodes.

*Job running:* Jobs will be running in the selected set of nodes with local scheduling. A node can either finish the job successfully or fail the job running because of node unavailability. If the system can support auto-rescheduling of failed jobs, the failed jobs can be rescheduled to run in other nodes in the same site for a certain number of times. The job is considered as failed only after it still cannot be finished after the allowed maximum number of rescheduling.

*Running result report:* The running results with the number of successful and failed jobs will be reported to the G-Scheduler after the running site stops the job running (either finished or failed). At the same time the site updates the intra-site trust based on the running result.



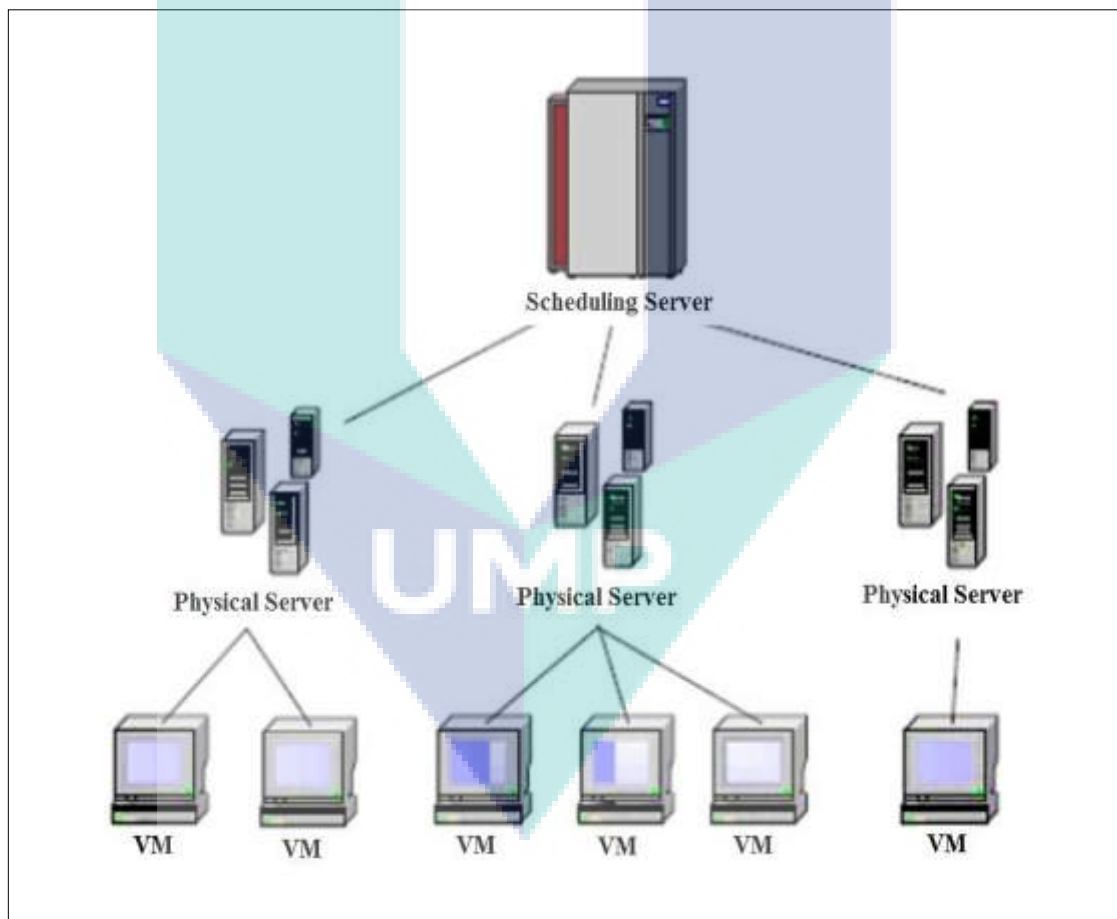
*Inter-site trust update:* The G-Scheduler updates the inter-site trust based on the reported running results (or perceived running results if the G-Scheduler owns the job running monitor function). If rescheduling is allowed, the failed jobs will enter the G-Queue again, and wait for the next global scheduling. This procedure continues until all jobs in one task have been successfully completed.



**Figure 2.10:** Jobs Scheduling Architecture C

Source: Liang and Shi, 2010

Shailesh (2011) describes the relationship between the physical machines and the VMs as shown in Figure 2.11. Consider  $P$  as a set of all the physical machines in the entire system, where  $P = \{P_1, P_2, P_3 \dots P_N\}$ .  $N$  is total number of the physical machines and an individual physical machine can be denoted as  $P_i$ , where  $i$  denote the physical machine number and range of  $i$  is  $(1 \leq i \leq N)$ . Similarly, we have a set of VMs on each physical machine  $P_i$ ,  $V_i = \{V_{i1}, V_{i2}, V_{im}\}$  here  $m$  is the number of VMs on the physical server  $i$  (Jinhua et al., 2011). If we want to deploy VM  $V$  on the present system, then we have a solution set denoted by  $S = \{S_1, S_2, S_3 \dots S_N\}$ , it represents the mapping solution after VM  $V$  is assigned to each of the physical machines. When the  $V$  is arranged with the physical machine  $P_i$  we get the mapping structure denoted as  $S_i$ .

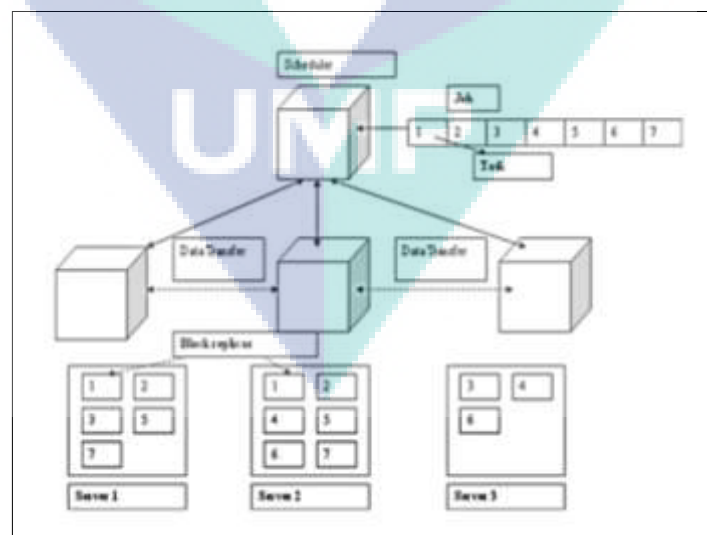


**Figure 2.11:** Cloud System Structure

Source: Shailesh 2011

(Vaishali et al., 2012) has introduced a data locality driven task scheduling algorithm, called Balance-Reduce. On finding a feasible solution, a critical obstacle is that the remote cost cannot be calculated before the remote task number is known. Moreover, it is hard to obtain a near optimal solution when the remote cost changes frequently. For example, when we allocate a remote task, the remote task number increase by one, so the remote cost may also increase. Furthermore, the load of the servers which have been allocated remote tasks must be updated. In order to make sure the remote cost, algorithm is split into two phases, balance and reduces:

- *Balance*: Given a data placement graph  $G$ , initial load set  $L_{init}$  and a local cost  $C_{loc}$ , the balance phase returns a total allocation  $B$ . Under  $B$ , all tasks are allocated to their preferred servers evenly.
- *Reduce*: Given a local cost  $C_{loc}$ , a remote cost function  $C_{rem}(\cdot)$ , a total allocation  $B$  computed by the balance phase, and an initial load set  $L_{init}$ , the reduce phase works iteratively to produce a sequence of total allocations and returns the best one. By taking advantage of  $B$ , the remote cost can be computed at the beginning of each iteration.



**Figure 2.12:** Cloud Task Scheduling Process

Source: Vaishali et al., 2012

### 2.4.1.2 Job Scheduling Architectures Summary

This section is summarizing the overall concept of above mentioned job scheduling architectures. Job scheduling architectures *A* (Nguyen and Lim, 2007), *B* (Ranganathan and Foster, 2003) and *C* (Liang and Shi, 2010) basically divide their schedulers into three phases. For the job submission by the users all three architectures follow First-Come-First-Serve (*FCFS*) scheduling policy. In job scheduler *A* and *B* phase 2 (*LS*- Local Scheduler) and phase 3 (*DS*-Data Scheduler) are identical, the only difference is in their phase 1. In architecture *A*, phase 1 is named as External Scheduler (*ES*) and in architecture *B* this phase is named as Resource Broker (*RB*). User jobs are submitted directly to *ES* and *RB* in both architectures where jobs are placed in queues. Both *ES* and *RB* have different scheduling policies at this level for job dispatch to *LS*. *ES* uses (i) load at the remote site and/or (ii) location of dataset as the resource scheduling strategy whereas *RB* uses estimation of cost in terms of time as the resource scheduling strategy. In Job scheduling architectures *C*, the architecture is also divided into three phases. But instead of *LS* and *DS* like in architecture *A* and *B*, it presents two *LS* with the Global Scheduler (*GS*). User jobs are submitted and queued to *GS*. *GS* uses stock of currency between the submission site and selected running sites based on the reputation information and resource request. *LS* works by using processor selection based on the intra-site trust information.

The scheduler picks the server first, which already has jobs being processed and waiting i.e. Local Scheduler (*LS*). The data begin to be transferred immediately means Data Scheduler (*DS*) will start its work. It is possible that all jobs for that server are finished by the time Execution Time (*ET*) that the transfer finishes so the time to enter service is merely the Data Transfer time (*DT*). It is also possible that the some jobs are not finished when the data transfer completes so that the new job must wait until all previous jobs are completed. That time is just the system time (*ET*) of the current jobs at the server. Thus the total time that the new job must wait before service is the Queue Time (*QT*) i.e. system time (*ET*) for all jobs currently at that server. That is why above architectures are taking maximum of *QT* and *DT* during the transfer time calculation (equation 1.1). According to these architectures the whole system is busy at a time, it

means that *DS* will keep continue its work and meanwhile there will be jobs in the *ES*, and *ES* will execute jobs and at the same time there will be jobs in *LS*.

Cheng (2007) proposed two schedulers MWTP and VWTP to perform proportional delay differentiation. Algorithms can maintain the delay proportion and reduce the average queuing delay by simultaneously considering the packet waiting time and the packet transmission time. (Liang and Shi, 2010) proposed a reputation-based resource scheduler for the Grid. (Dwekat and Rouskas, 2011) presented tiered-service fair queuing (TSFQ) scheduler techniques, within each tier, the schedulers employ a fixed number of queues to handle packets with few or no sorting operations. (Francini et al., 2001) have presented three enhancements of WRR schedulers for providing bandwidth guarantees in IP networks.

In addition, all these scheduling architectures work on the base of Queuing theory. Each scheduler technique has its own architecture and various numbers of sub-schedulers. On the bases of above discussed scheduler architectures it means that, the number of scheduler and the combination of schedulers depends on the scheduler architecture.

#### **2.4.2 QUEUING THEORY**

Queuing theory and models are just like the scheduling backbones. Here in this section queuing models and statistical distributions will be briefly discussed. As the value of a model fluctuates with its outcomes, suitable models and algorithms need to be selected. Another significant factor is the point of view taken. Performance values calculated with respect to an arriving process are not essentially equal as these processes are determined from a server's viewpoint (Stallings, 2000). Again, the impact of statistical distributions is not minor to be neglected (Tang et al., 2006). Though, it turns out that the performance significance is the same, during the use of models with exponentially distributed inter-arrival and service times. On the other hand, a lot of beneficial relations have been determined for more common cases. Queuing theory with

models have briefly described by (Introduction to Queuing Theory, 2011; Lec-30, 2010; Lec-31, 2010; Lec-32, 2010; and Queuing Theory-Birth Death processes, 2011). Though queuing models differ in application and complexity, a common set of performance features may be determined as follows.

In order to calculate all related parameters' values, we need to use the following parameters. For each model, by using various values of Arrival Rate ( $\lambda$ ), Service Time Distribution ( $\mu$ ), Server Utilization ( $\rho$ ), Number of Servers ( $C$ ), Number of Jobs or Population ( $M$ ), Queue Capacity ( $K$ ), Bandwidth ( $BW$ ), we need to calculate the following values, which are the key parameters while calculating Total Completion Time for transferring.

$P_n$  = Probability of 'n' jobs in the system

$P_0$  = Probability of '0' jobs in the system

$L$  = Average jobs in the system

$L_q$  = Average jobs in the queue

$W_s$  = Average time spent in the system

$W_q$  = Average time waiting in line

The state probability  $p_n$  is defined through the probability of  $n$  jobs residing in the system, either being served or waiting. Thus,  $p_n = \{n \text{ jobs in the system}\}$  has giving by (Intro to Queuing Theory, 2011; and Queuing Theory-Birth Death processes, 2011). The flow intensity  $\rho$  is given by the ratio of arrival rate  $\lambda$  and service rate  $\mu$ , i.e.

$$\rho = \frac{\lambda}{\mu} \quad (2.1)$$

Queuing theory dealing with mathematical techniques used to analyse congestion problems. According to (Mag et al., 2009; and Lipsky, 2009), congestion may occur when a population of entities (jobs) has to share a service system with limited capacity. Every time there are more jobs requiring service than they can be attended to these jobs are said to form a queue or waiting line. The main components of

a queuing system are the entities that require service, often called *jobs*; the entities that provide service, usually called *servers*, and one or more *queues* discussed by (Mag et al., 2009; Lipsky, 2009; Chee-Hock and Soong, 2008; and Jain et al., 2007). Jobs are often external to the service system. They individually decide when they need a certain service. Then they are said to arrive at the service system. Queues may build up and dissolve again from time to time even if the capacity of the service system is sufficiently large to handle all service demands in the long run. This phenomenon leads to the formulation of stochastic models discussed by (Philippe, 1998; and John et al., 2008).

Throughout this study, queue models refer to the total number of jobs present in a service centre (*server*), including the jobs which are in service, and waiting time refers to the time jobs waiting for their service, excluding their service time. There are some notations used for the presentation of parameters in queuing models.

#### 2.4.2.1 Kendall's classification

Service systems with a single queue are conventionally classified according to Kendall's notation which has been discussed by (Zafril and Azmi, 2011; Mag et al., 2009; Lipsky, 2009; Chee-Hock and Soong, 2008; John et al., 2008; Dattatreya, 2008; Artalejo and Lopez, 2007; Jain et al., 2007; and Gupta and Samanta, 2004).

#### Notation A/B/C

The symbol at position *A* describes the *arrivals* to the queue process, often in terms of an inter-arrival time distribution. *A* denotes the passive time distribution of the individual job for systems with a finite job population. The symbol at position *B* signals the *service* time distribution. The integer at position *C* shows the service *capacity* (number of servers). It is customary to indicate an exponential distribution by the symbol *M* for Markovian or memory less. Service times are generally assumed to be mutually independent and independent of the arrival process. A prefixed *B* at position *A* or *B* indicates that arrivals or services occur in batches (groups). Without this prefix, jobs are assumed to arrive at one by one and to be served individually.

Occasionally, this organization is extended to a four, five, or six symbol form. Each omitted component has a default value. The complete form of Kendall's classification is as: A/B/C/D/E/F.

### **Notation A/B/C/D/E/F**

The symbols at positions *A*, *B*, and *C* are the same, as discussed above. The integer at position *D* denotes the capacity of the queue; that is, the maximum *number of jobs*, which can be simultaneously accommodated in the system (default: *infinite*). The integer at position *E* depicts the size of the *job population* (default: *infinite*). The symbol at position *F* indicates the service *discipline* (default: *FCFS*), the symbol *GD* at the last position shows a general (*unspecified*) service discipline.

## **2.4.3 Queuing Models**

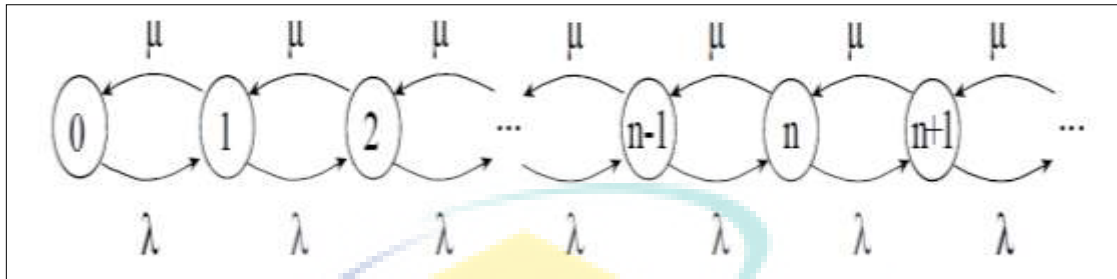
Queuing theory consists of various models, depending on the number of servers (single or multiple), queue length (fixed, dynamic) and population of jobs (known or unknown) as well as on the basis of scheduling discipline. Queuing Models have been briefly discussed by (Queuing Theory, 2011; Intro to Queuing Theory-Birth Death processes, 2011; Lec-30, 2010; Lec-31, 2010; and Lec-32, 2010).

### **2.4.3.1 M/M/1 Model**

M/M/1 is the most basic model in the queuing theory. In this section, the mathematical operation will be combined with intuitive insights to prepare the path for more composite models. In M/M/1 model, random arrivals and exponentially distributed service times are presumed. The random arrivals are exactly defined to be Poisson in statistics. In addition, there is only a single server serving jobs on a First-Come-First-Serve discipline basis. The arriving jobs are unaffected by the queue size because the population is infinite. Using parameters in the M/M/1 model are  $\lambda$  (average



arrival rate) and  $\mu$  (average service rate). Figure 2.13 shows the state transition diagram for single server system.

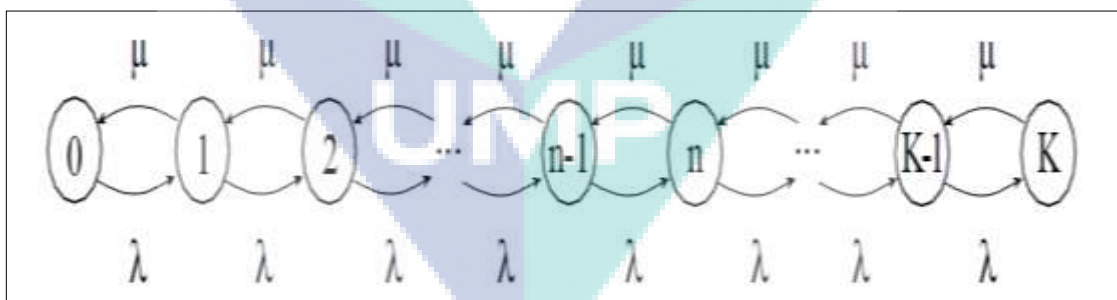


**Figure 2.13:** *M/M/1* State transition diagram

Source: Queuing Theory 2011

#### 2.4.3.2 *M/M/1/K* Model (Capacity Constraint)

Subsequently, the next step is to spread the *M/M/1* model to include a system capacity constraint (limitation) thus becomes *M/M/1/K* model (Lec-30, 2010; and Intro to Queuing Theory, 2011). Same condition described for state 0 before has to be applied to state *K* as well. As shown in Figure 2.14.



**Figure 2.14:** *M/M/1/K* Model State transition diagram

Source: Queuing Theory 2011

### 2.4.3.3 M/M/c Model (Multi-server System)

Multiple servers in queuing systems may be modeled by a single server system with state dependent service rate. Given  $n$  jobs are in the system, work is processed  $n$  times as quick as a single server would require to do so. Given a limited provision of servers, the load dependent service rate will be the same. Queuing Models briefly have been described by (Lec-30, 2010; Lec-31, 2010; Lec-32, 2010; Intro to Queuing Theory, 2011; and Queuing Theory-Birth Death processes, 2011). The related model is denoted as M/M/c in the limited case and M/M/ $\infty$  in the unlimited case. The latter system is also called '*delay server*', as the average response time is insensible to the number of jobs presented in the system. As a single system, the delay server is almost useless, but if combined with other systems to a queuing network, it plays a significant role.

### 2.4.3.4 M/M/c/K Model (Capacity Constraints with Multi-server Systems)

As mentioned above, by customizing the parameters for the load dependent model, the capacity constraints can be introduced to a multi-server system M/M/c/K, as briefly discussed by (Lec-30, 2010; Lec-31, 2010; Lec-32, 2010; and Queuing Theory-Birth-Death processes, 2011).

### 2.4.3.5 M/M/c\*/P Model (Finite Population)

Earlier discussion has centred on the queuing theory with infinite job population. Although mathematically convenient, such a supposition only serves well as an approximation to situations with a large population (number of jobs). One anticipates that prediction faults become negligible. If this is not the case, then one has to keep an eye on finiteness. This is best done by modifying the birth rate  $\lambda$  in the standard model birth-death shown as follows:

$$\lambda_n = \begin{cases} (M - n) \lambda & \text{for } 0 \leq n < M \\ 0 & \text{for } n \geq M \end{cases} \quad (2.2)$$

Here  $M$  denotes the number of jobs (size of the population). Presume a system with  $c < M$  service units, i.e.

$$\mu_n = \begin{cases} n\mu & \text{for } 1 \leq n < c \\ c\mu & \text{for } n \geq c \end{cases} \quad (2.3)$$

$$p_n = \begin{cases} \binom{M}{n} \rho^n p_0 & \text{for } 0 \leq n < c \\ \binom{M}{n} \frac{n!}{c^{n-c} c!} \rho^n p_0 & \text{for } c \leq n \leq M \end{cases} \quad (2.4)$$

with  $\binom{M}{n} = \frac{M!}{(M-n)!n!}$  denoting the binomial coefficient. Using  $\sum_{n=0}^M p_n = 1$  and solving for  $p_0$  gives

$$p_0 = \left[ \sum_{n=0}^{c-1} \binom{M}{n} \rho^n + \sum_{n=c}^M \binom{M}{n} \frac{n!}{c^{n-c} c!} \rho^n \right]^{-1} \quad (2.5)$$

For effective calculation of the steady state probabilities, (Gross and Harris, 1985; and Gross, 2008) propose the following recursion based on the properties of the binomial coefficient:

$$f_n = \frac{p_{n+1}}{p_n} = \begin{cases} \frac{M-n}{n+1} & \text{for } 0 \leq n < c \\ \frac{M-n}{c} & \text{for } c \leq n \leq M \end{cases}, \quad p_n = \prod_{i=0}^{n-1} f_i p_0 \quad (2.6)$$

Using the definition of the expected result, one is now able to calculate the average system size:

$$L = \sum_{n=0}^M n p_n = \left[ \sum_{n=0}^{c-1} \binom{M}{n} \rho^n + \sum_{n=c}^M \binom{M}{n} \frac{n!}{c^{n-c} c!} \rho^n \right] p_0 \quad (2.7)$$

To follow the work of (Gross and Harris, 1985; Lec-30, 2010; Lec-31, 2010; Lec-32, 2010; Intro to Queuing Theory, 2011; and Queuing Theory-Birth Death processes, 2011), the average queue size  $L_q$  can be derived from  $L$  as follows:

$$\begin{aligned}
 L_q &= \sum_{n=c}^M (n-c) p_n = \sum_{n=c}^M n p_n - c \sum_{n=c}^M p_n \\
 &= L - \sum_{n=c}^{c-1} n p_n - c \left( 1 - \sum_{n=0}^{c-1} n p_n \right) \\
 &= L - c + \sum_{n=0}^{c-1} (c-n) p_n \\
 &= L - c + p_0 \sum_{n=0}^{c-1} (c-n) \binom{M}{n} \rho^n \tag{2.8}
 \end{aligned}$$

In order to derive the waiting time indicators using Little's law, one initially has to define the mean arrival rate  $\bar{\lambda}$  (Lec-31, 2010; Lec-32, 2010; Queuing Theory-Birth Death processes, 2011). With  $n$  jobs already present in the system and a maximum of  $M-n$  jobs remain outside waiting for their turn. This outcomes in a mean arrival rate of  $(M-n)\lambda$ , averaging yields

$$\begin{aligned}
 \bar{\lambda} &= \sum_{n=0}^M (M-n) \lambda p_n = \lambda (M \sum_{n=0}^M p_n - \sum_{n=0}^M n p_n) \\
 &= \lambda (M - L) \tag{2.9}
 \end{aligned}$$

By using Little's law with the just calculated mean arrival rate  $\bar{\lambda}$  leads to

$$W_s = \frac{L}{\lambda(M-n)} \tag{2.10}$$

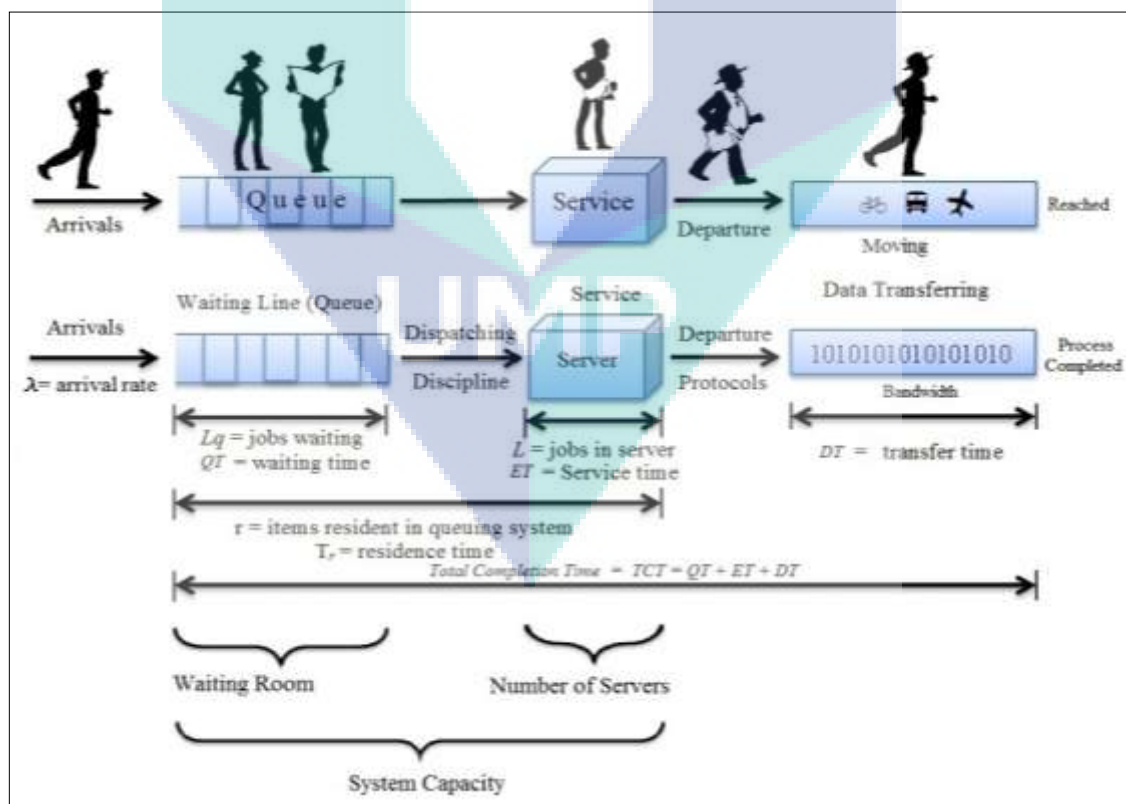
and

$$W_q = \frac{L_q}{\lambda(M-n)} \tag{2.11}$$

With the large number of population  $M$ , the Engset distribution approaches the probabilities where  $p_n$  is given by the M/M/c/c Erlang loss system. (Gross and Harris, 1985; Lec-30, 2010; Lec-31, 2010; Lec-32, 2010; and Intro to Queuing Theory, 2011) have discussed in great detail about this queuing theory.

#### 2.4.4 Queue Characterization

A queuing system may be termed as a system, where jobs arrive conferring to an arrival process to get service by a service facility according to a service process. Each service facility may have one or more servers. It is normally assumed that each job can get service only from one server at a time. If all servers are busy, the job has to put in the queue for service. When a server gets free time again, the next job is selected from the queue according to the instructions given by the queuing discipline. The job might run through one or more stages during the service, before the departure from the system. Figure 2.15 describes the flow process of a job in queue system.



**Figure 2.15:** Schematic representation of a queuing system

When a job needs to transfer from one node to another, first, it has to join the queue. In the queue, the job should wait for its turn. Here, waiting time in the queue is denoted by  $QT$ . This waiting time starts immediately after entering the queue and just before starting the execution process. The number of job in the queue in a specific time is denoted by  $Lq$ . After waiting, the job will enter to the processor. During execution, the time taken by the processor is denoted by  $ET$ .  $ET$  starts with the start of execution and ends with the end of execution. The number of jobs inside the processor in a specific time is denoted by  $L$ . Resident time is  $Tr$ , the time which a job passes in the system, which is the sum of the queue time ( $QT$ ) and the execution time ( $ET$ ). After completion of job execution, the transfer process will start. Transfer time denoted by  $DT$ , which commences with the start of the transfer process and ends with reaching the job to its destination. Figure 3.6 in chapter number 3 shows that Total Completion Time ( $TCT$ ) is the sum of the waiting time in the queue ( $QT$ ), spend time in the system ( $ET$ ), and transferring time ( $DT$ ). A schematic representation of such system is shown in Figure 2.15. Before moving forward, the most significant aspects of queuing systems are listed and described.

The arrival process is stated by a statistical distribution and its parameters. Normally, the exponential distribution is assumed resulting in the arrival pattern to calculate the average number of arrivals per unit of time. Generally, arrival processes are characterized by other configurations as well. These include batch arrivals and time dependence.

- The service process is the same as the arrival process. Again, exponentially is normally assumed in practice due to intractability when releasing these assumptions. The service process is greatly dependent on the state of the system which is the opposite of the arrival process. In the case of the queuing system being empty, the service facility will become idle.
- The queuing discipline refers to the way, jobs selected for the service under queuing situations. It is generally used and the most common is the First-

Come-First-Serve (*FCFS*) discipline. Others include Last-Come-First-Serve (*LCFS*), random and priority service.

- The departure process is rarely used to define a queuing system, as it can be seen as an outcome of the queuing discipline, arrival, and service process. Under certain circumstances, arrival and departure process follows the same statistical distribution, which has become a very significant fact in the queuing network modelling.
- The number of server's states to the number of parallel nodes, which can service jobs simultaneously at a time.
- The number and organization of service stages, a job which might have to visit before departing from the system. Shortly in a computer system, a job might have to visit the CPU twice and the Input/Output processor once during a single service.

## 2.5 SUMMARY

This chapter briefly reviews and discusses Cloud Computing, Cloud services, their advantages and disadvantages, types of Clouds, capabilities, and also compares the prices of various Cloud services. Data Management is the key factor in providing Cloud services. Finally, data-management scheduling, service discipline queuing theory, and queuing models are also presented.

To calculate data transfer time from source to target, we need to calculate the Queue Time, Execution Time, and Data Transfer Time. We have to calculate Length of the queue ( $Lq$ ), wait in the queue for a job ( $QT$ ), number of jobs in the server ( $L$ ), and wait in the server for a job ( $ET$ ). Similar the scheduling architecture is encapsulated in three distinct modules Figure 2.8, Figure 2.9 and Figure 2.10.

*External Scheduler (ES):* Each user in the system is associated with an External Scheduler and submits jobs to that External Scheduler. We can typically imagine one *ES* per site but the framework does not enforce this. The *ES* then decides the remote site to which to send the job to depending on some scheduling algorithm. It may use external information such as load at a remote site or the location of a dataset, as input to its decisions.

*Local Scheduler (LS):* Once a job is assigned to run at a particular site (and sent to an incoming job queue of that site) it is managed by the Local Scheduler at that site. The *LS* decides how to schedule all jobs allocated to it, on its local resources. It could, for example, decide to give priority to certain kinds of jobs, or it could refuse to run jobs submitted by a certain user.

*Dataset Scheduler (DS):* The *DS* at each site keeps track of the popularity of each data set locally available. It then replicates popular datasets to remote sites depending on some algorithm. The *DS* may use external information such as whether the data already exists at a site and load at a remote site to guide its decisions.

Next Chapter will discuss the methodology that is, how we can get all these *Queue Time (QT)*, *Execution Time (ET)*, and *Data Transfer Time (DT)*, *Length of the Queue (Lq)*, *number of jobs in the server (L)*. A queuing model, the finite population *M/M/c/K* model will be used to evaluate the new proposed scheduling model.



## CHAPTER 3

### METHODOLOGY

#### 3.1 INTRODUCTION

This chapter introduces an alternate technique to calculate the time for data transfer from source to target. To improve data efficiency and reliability, the current chapter uses various mathematical formulae. A scheduling architecture will be presented, together with the Queuing Models, and their use will be discussed.

#### 3.2 QUEUING THEORY AND SCHEDULING

According to (Stallings, 2000; Intro to Queuing Theory, 2011; and Queuing Theory-Birth Death processes, 2011), Equation 3.1 illustrates some important parameters associated with a queuing model. Items arrive with some average rates (i.e. items arriving per second). Here, just for understanding, Little's Law notations have been used. At any given time, a certain number of items will be waiting in the queue (zero or more); assuming the average number of items waiting is  $W$ , and the meantime that an item must wait is  $T_w$ . The  $T_w$  is an average over all incoming items, including those that do not wait at all. The server handles incoming items with an average service time  $T_s$ ; this is the time interval between the dispatch of an item to the server and the departure of that item from the server. Finally, two parameters apply to the system as a whole. The average number of items resident in the system, including the item being

served (if any) and suppose the items waiting (if any), is  $T_w$ ; and the average time that an item spends in the system, waiting and being served, is  $T_s$ ; we refer this as the mean residence time. First-In-First-Out (*FIFO*) is a suitable scheduling discipline to use.

### 3.2.1 Parameters Used In Queuing Theory

This study is using general parameters/notations. All parameters have been classified into two categories, namely *independent* and *dependent*. Some notations have been changed from Little Law notations to general notation used by (Tang et al., 2006; Ranganathan and Foster, 2003; and Nguyen and Lim, 2007) to avoid confusion. Throughout this research, the *Average time spent in the system* is denoted by  $ET$  instead of  $W_s$ , *Data Transfer Time* is represented by  $DT$  instead of  $T_t$  and *Average time waiting in queue* denoted by  $QT$  instead of  $W_q$ . The rest of the parameters are the same, as they have been presented in previous works.

#### 3.2.1.1 Independent Parameters

Independent parameters are those which are not dependent on other parameters. Independent parameters can be put at the first time. In this study, the followings are independent parameters:

$\lambda$  = Arrival Rate

$\mu$  = Service Time Distribution

$\rho = \frac{\lambda}{\mu}$  = Server Utilization

$C$  = Number of Servers

$M = P$  = Number of Jobs (Population)

$K$  = Queue Capacity

$BW$  = Bandwidth

### 3.2.1.2 Dependent Parameters

Dependent Parameters are those parameters which are depended on other parameters. Dependent parameter's values are generated by using independent parameters. In this study, the followings are dependent parameters.

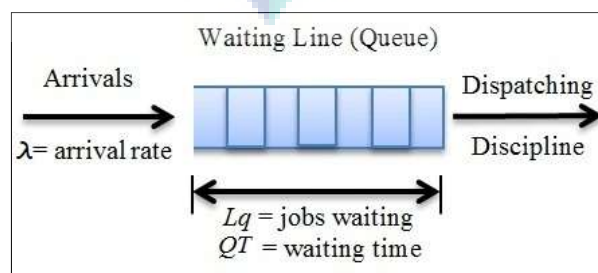
$P_n$  = Probability of 'n' jobs being in the system  
 $P_0$  = Probability of 'o' jobs being in the system  
 $L$  = Average jobs in the system  
 $Lq$  = Average jobs in the queue  
 $ET = W_s$  = Average time spent in the system  
 $QT = W_q$  = Average time waiting in line (queue)  
 $DT = T_t$  = Data Transfer Time

### 3.2.1.3 Parameters Definition and Explanation

For better and easy understanding, each parameter will be defined. Some of them belong to Little's Law. It is important in scheduling and queuing theory. Little Law (Stallings, 2000) has been discussed by (Blanc, 2011; Lipsky, 2009; and Chee-Hock and Soong, 2008) as follows:

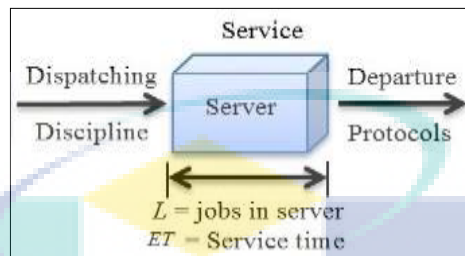
$$T_r = T_w + T_s \quad (3.1)$$

Where  $T_w$  is the mean waiting time generally denoted by  $QT$ , which the time for a job is passing in waiting after entrance to queue and before starting execution, as shown in Figure 3.1.



**Figure 3.1: Time in Queue**

Mean service (execution) time  $T_s$  denoted by (Little et al., 2008) is for each arrival;  $ET_{j,i}$  denoted by (Nguyen and Lim, 2007; Tang et al., 2006; and Ranganathan and Foster, 2003).  $ET$  is the job *Execution Time* after entrance from queue and before starting the transfer process, shown in Figure 3.2.



**Figure 3.2:** Time in Server

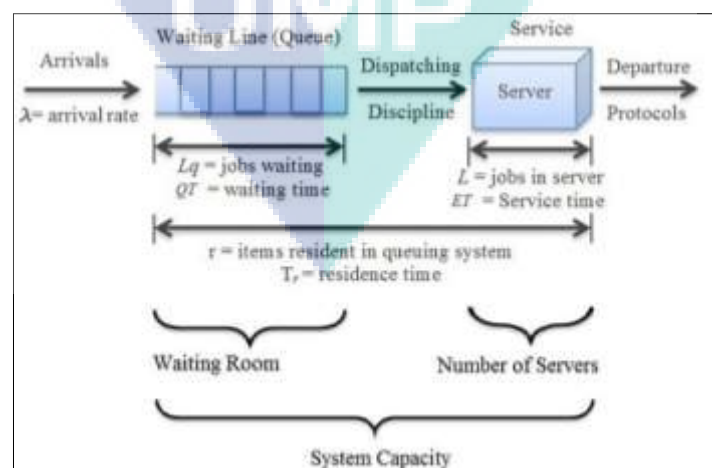
$Tr$  (residence time) given in Equation 3.1 is the meantime, an item spends in the system means both times in *queue* as well as in *server*. Hence using Little's Law,  $Tr$  has shown in Figures 3.3 and 3.4.

$$Tr = Wq + Ws$$

$$Tr = QT_{(i)} + ET_{j,i}$$

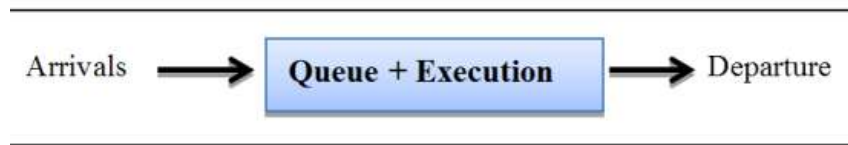
(3.2)

Where  $QT_{(i)} = Wq$  and  $ET_{j,i} = Ws$  as mentioned in Figure 3.3.



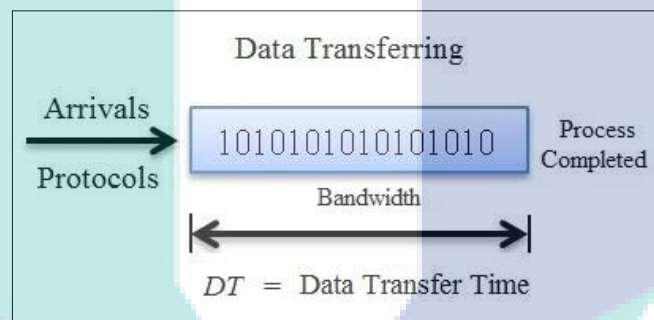
**Figure 3.3:** Schematic representation of time in system

Or  $Tr$  is the sum of  $QT$  and  $ET$ , as described in Figure 3.4.



**Figure 3.4:** Time in System

The  $DT_{j,i}$  is the time after the completion of execution and before the completion of data transfer process, as presented by  $Tt$ . The important point is that  $Tt$  is only the transfer time excludes  $QT$  and  $ET$ , as shown in Figure 3.5.



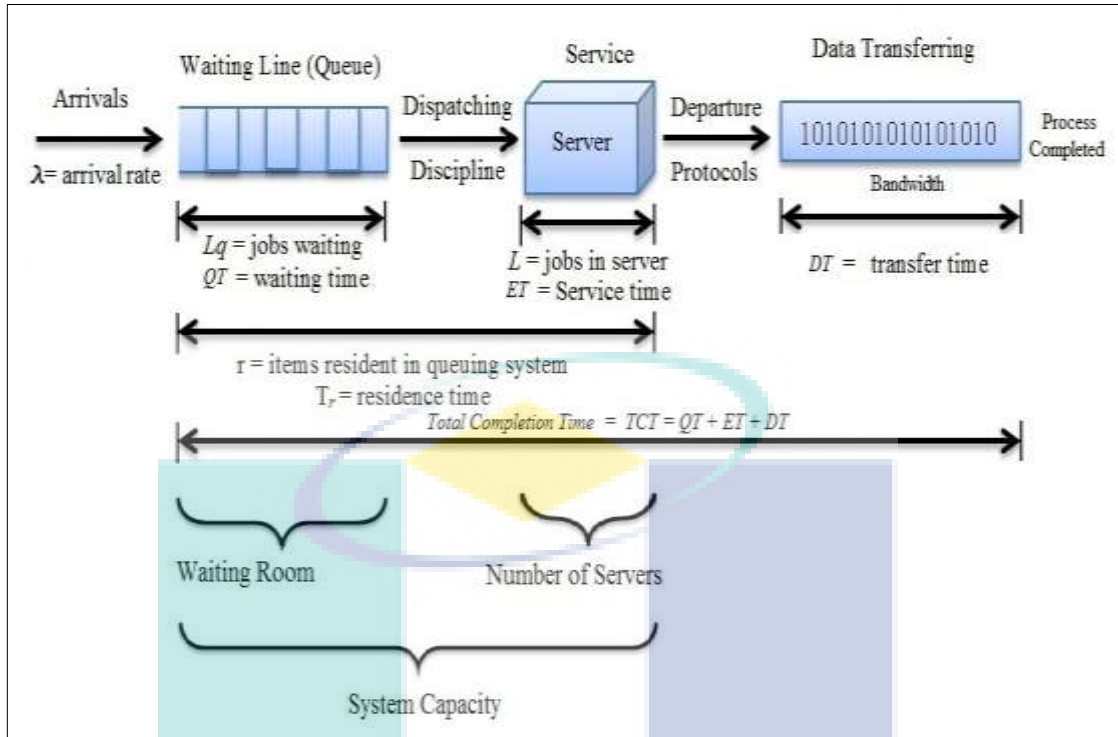
**Figure 3.5:** Transferring Time

According to this study, the Total Completion Time ( $TCT$ ) for a job can be the sum of three different Times,  $QT$ ,  $ET$  and  $DT$ , as shown in Figure 3.6.

$$TCT_{j,i} = QT + ET + DT(f(j)) \quad (3.3)$$

Converting this expression into the Little's notation, the Equation 3.3 will be as:

$$TCT_{j,i} = Wq + Ws + Tt(f(j)) \quad (3.4)$$

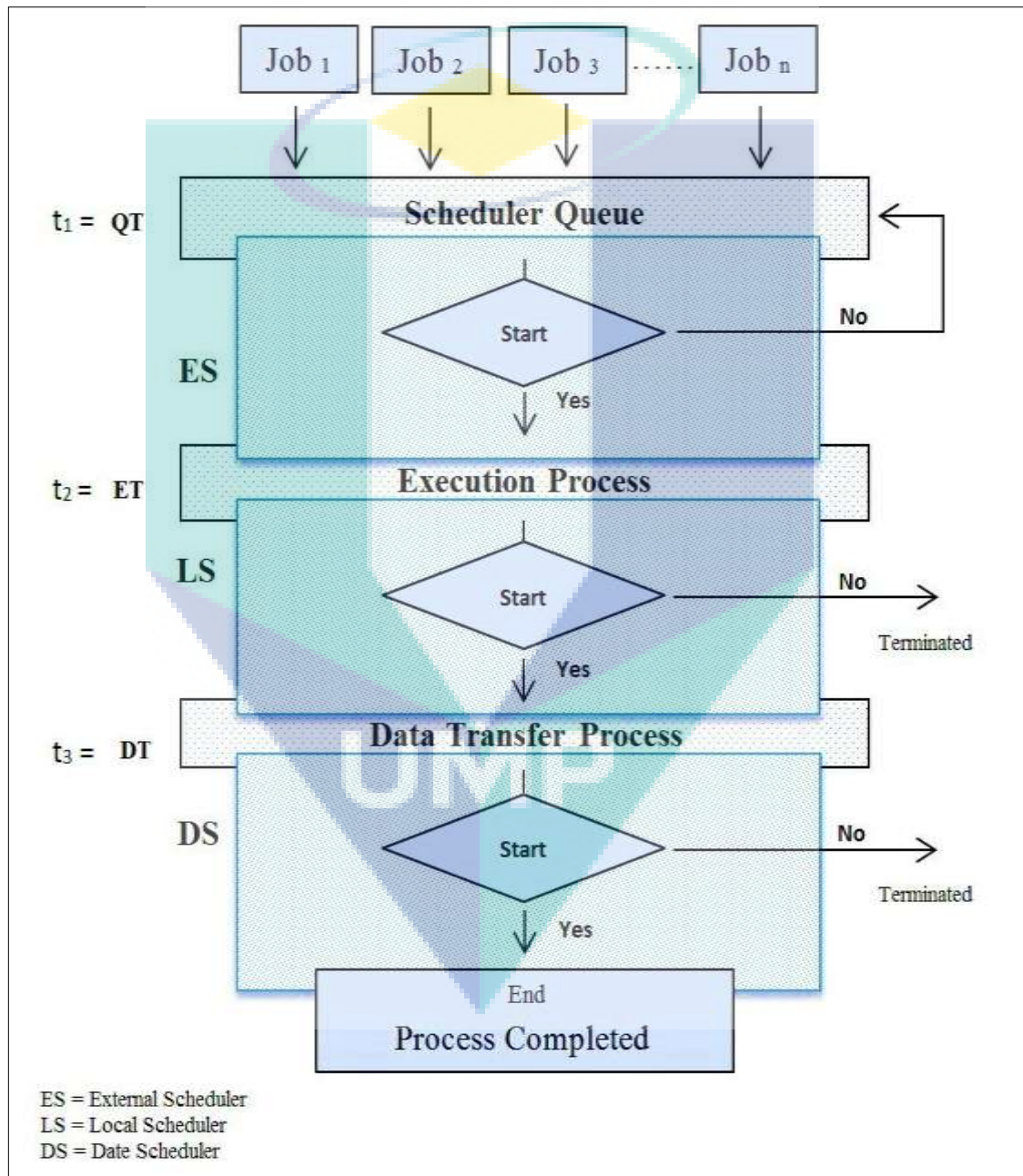


**Figure 3.6:** Total Completion Time for Data Transfer

Equation 3.3 has explained in Figure 3.6. When a job needs to transfer from one node to another, first, it has to join the queue. In the queue, the job should wait for its turn. Here, this waiting time in the queue is denoted by  $QT$ . This waiting time starts immediately after entering the queue and just before starting the execution process. The number of job in the queue in a specific time is denoted by  $Lq$ . After waiting, the job will enter the processor. During execution, the time taken by the processor is denoted by  $ET$ .  $ET$  starts with the start of execution and ends with the end of execution. The number of jobs inside the processor in a specific time is denoted by  $L$ . Resident time is  $T_r$  which a job passes in the system, and is the sum of the queue time ( $QT$ ) and the execution time ( $ET$ ). After completion of job execution, the transfer process will start. Transfer time denoted by  $DT$ , which commences with the start of the transfer process and ends with reaching the job to its destination. Figure 3.6 shows that Total Completion Time ( $TCT$ ) is the sum of the waiting time in the queue ( $QT$ ), spend time in the system ( $ET$ ), and transferring time ( $DT$ ).

### 3.2.2 SCHEDULER ARCHITECTURE

Total Completion Time ( $TCT$ ) has three intervals of *times*; just like that the process flow consists of the scheduling technique also has three distinct modules, as shown in Figure 3.7.



**Figure 3.7:** Scheduler Architecture

*External Scheduler (ES):* Each user is associated with an External Scheduler in the system and submits jobs to that External Scheduler. *ES* decides the remote site to which the job will send to depending on some scheduling algorithms. *ES* uses external information to decide the input such as loading at a remote site or on a specific location of a data set.

*Local Scheduler (LS):* Assigned jobs are managed by the Local Scheduler to run at a particular site. The allocation of allocated jobs is also the responsibility of the *LS*. *LS* decide the priority and can refuse to run jobs submitted by a certain user.

*Dataset Scheduler (DS):* At each site, *DS* keeps track of the popularity of each local available data set. It can use external information, such as the already existed data at the concern site, to load to the target remote site.

Job scheduling architectures A (Nguyen and Lim, 2007), B (Ranganathan and Foster, 2003) and C (Liang and Shi, 2010) basically divide their schedulers into three phases. For the job submission by the users all three architectures follow First-Come-First-Serve (FCFS) scheduling policy. In job scheduler A and B phase 2 (*LS*- Local Scheduler) and phase 3 (*DS*-Data Scheduler) are identical, the only difference is in their phase 1. In architecture A, phase 1 is named as External Scheduler (*ES*) and in architecture B this phase is named as Resource Broker (*RB*). User jobs are submitted directly to *ES* and *RB* in both architectures where jobs are placed in queues. Both *ES* and *RB* have different scheduling policies at this level for job dispatch to *LS*. *ES* uses (i) load at the remote site and/or (ii) location of dataset as the resource scheduling strategy whereas *RB* uses estimation of cost in terms of time as the resource scheduling strategy. In Job scheduling architectures C, the architecture is also divided into three phases. But instead of *LS* and *DS* like in architecture A and B it presents two *LS* with the Global Scheduler (*GS*). User jobs are submitted and queued to *GS*. *GS* uses stock of currency between the submission site and selected running sites based on the reputation information and resource request. *LS* works by using processor selection based on the intra-site trust information.

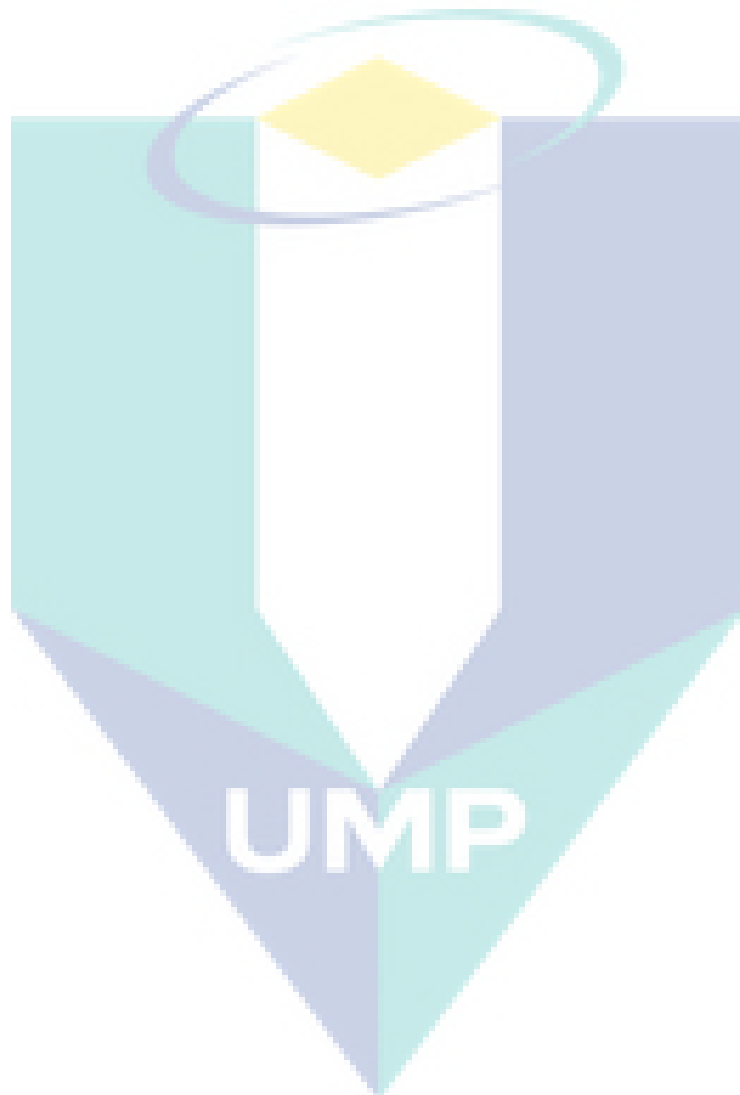


Cheng (2007) proposed two schedulers MWTP and VWTP to perform proportional delay differentiation. Algorithms can maintain the delay proportion and reduce the average queuing delay by simultaneously considering the packet waiting time and the packet transmission time. (Liang and Shi, 2010) proposed a reputation-based resource scheduler for the Grid. (Dwekat and Rouskas, 2011) presented tiered-service fair queuing (TSFQ) scheduler techniques, within each tier, the schedulers employ a fixed number of queues to handle packets with few or no sorting operations. (Francini et al., 2001) have presented three enhancements of WRR schedulers for providing bandwidth guarantees in IP networks.

The scheduler picks the server first, which already has jobs being processed and waiting i.e. Local Scheduler (*LS*). The data begin to be transferred immediately means Data Scheduler (*DS*) will start its work. It is possible that all jobs for that server are finished by the time Execution Time (*ET*) that the transfer finishes so the time to enter service is merely the Data Transfer time (*DT*). It is also possible that the some jobs are not finished when the data transfer completes so that the new job must wait until all previous jobs are completed. That time is just the system time (*ET*) of the current jobs at the server. Thus the total time that the new job must wait before service is the Queue Time (*QT*) i.e. system time (*ET*) for all jobs currently at that server. That is why above architectures are taking maximum of *QT* and *DT* during the transfer time calculation (equation 1.1). According to these architectures the whole system is busy at a time, it means that *DS* will keep continue its work and meanwhile there will be jobs in the *ES*, and *ES* will execute jobs and at the same time there will be jobs in *LS*.

With encapsulation of three schedulers *ES*, *LS*, and *DS*, the time for completion of a job is also the encapsulation of three *Times* (Intervals). *QT* is the time when a job passes in waiting after entrance to queue and before starting execution. The time denoted by *ET* is the job execution time after its entrance from the queue and before starting the transfer process. *DT* is the time after completion of the execution time and before completion of the data transfer process. The Total Completion Time for a job is the sum of all these three *times*.

In addition, all these scheduling architectures work on the base of Queuing theory. Each scheduler technique has its own architecture and various numbers of sub-schedulers. On the bases of above discussed scheduler architectures, it means that, the number of scheduler and the combination of schedulers depends on the scheduler architecture.



### 3.2.3 SCHEDULING ALGORITHM FOR TCT

#### Algorithm's Parameters

Filters: ES; LS; DS.

Filtering  $\in$  ES  $\cap$  LS  $\cap$  DS.

Processing Time:  $t1(QT) + t2(ET) + t3(DT)$

*ES=External Scheduler*

*LS=Local Scheduler*

*DS=Data Scheduler*

*QT=Queue Time*

*ET=Execution Time*

*DT=Data Transfer Time*

#### The Algorithm: Total Completion Time (TCT) Algorithm

```

1: jobs submitted to scheduler queue for processing
2: for all jobs there is a QT.
3: end for
4:   if filter  $\notin$  ES (condition: job size  $\neq$  1 kb)
5:     job status: QT
6:     loop (go to 3)
7:     until: ES (condition: job size = 1 kb)
8:     end loop
9:     for all jobs there is an ET.
10:    end for
11:    if filter  $\in$  ES (condition: job size = 1 kb) &
        Filter  $\notin$  ES  $\cap$  LS (condition for LS =
        resources unavailable).
12:    job status: terminated
13:    end if
14:    if filter  $\in$  ES  $\cap$  LS (condition: job size = 1kb
        & resources available)
15:      for all jobs there is a DT.
16:      end for
17:      filter  $\notin$  ES  $\cap$  LS  $\cap$  DS (condition for DS=
        network congestion)
18:      job status: terminated
19:    else
20:      filter  $\in$  ES  $\cap$  LS  $\cap$  DS
21:      TCT = QT + ET + DT.
22:      job status: processed
23:    end if
24:  end if

```

### 3.2.3.1 ALGORITHM'S DESCRIPTION

Algorithm 3.2.3 encapsulates three schedulers, *ES* (*External Scheduler*), *LS* (*Local Scheduler*), and *DS* (*Data Scheduler*), as well as the time for completion of a job is also the encapsulation of three *Times* (Intervals), *QT* (*Queue Time*), *ET* (*Execution Time*), and *DT* (*Data Transfer Time*). *QT*, *ET* and *DT* are denoted by  $t_1$ ,  $t_2$  and  $t_3$  respectively. All these parameters have described in Figure 3.7. All jobs, which we need to transfer, submitted to scheduler's Queue. All schedulers filter jobs according to the filter's conditions, and then process.

Each user is associated with an External Scheduler in the system and submits jobs to the External Scheduler's Queue. *ES* decides the remote site to which the job will send to depending on some scheduling algorithms conditions. *ES* uses external information to decide the input such as loading at a remote site or on a specific location of a data set. *ES* filtering condition that is one job must be 1 kb. After fulfilling the condition, *ES* will keep jobs in *QT*.

After the completion of *ES* process, jobs assign and manage by the Local Scheduler to run at a particular site. The allocation of allocated jobs is the responsibility of *LS*, as well as decides the priority and can refuse to run jobs submitted by a certain user. Time consumed by *ES* is actually execution time and denoted by *ET*. *ES* will check, either resource for transferring available, to send job or put in waiting queue until the availability of resources.

When *ES* completes its responsibility of checking the availability of resources, jobs assigns to *DS*. *DS* check the network congestion; either network is able for transferring or *DS* should keep jobs in waiting queue until the availability of network. At each site, *DS* keeps track of the popularity of each local available data set. It can use external information, such as the already existed data at the concern site, to load to the target remote site. Consumed time during the process of *DS* is denoted by *DT*. According to the new proposed technique the sum of all intervals ( $t_1$ ,  $t_2$  and  $t_2$ ) give TCT, which is the Total Completion Time for transferring of data.

### 3.2.4 SCHEDULING STRATEGY

The resource scheduling strategy is based on the estimation of cost (in terms of time cost) of executing a job in each site. It is possible to assume that job is submitted to Local Scheduler (*LS*) one by one. When receiving a job submission, the *LS* will estimate the Total Time (*TT*) for completing a job in a site *i*, as mentioned by (Nguyen and Lim, 2007; Tang et al., 2006; and Ranganathan and Foster, 2003), as shown below.

$$TT_{k,i} = \max\{QT_{(i)}, DT(f(k),i)\} + ET_{k,i} \quad (3.5)$$

Scheduler's architectures have been presented by (Dwekat and Rouskas, 2011; Liang and Shi, 2010; Cheng, 2007; and Francini et al., 2001) with various structures, algorithms and with different number of schedulers in different numbers of slots. It depends on the way, how they have defined their model. Assume that we cannot begin the downloading, means data transferring process, until after the server is clear of the other jobs. For *Turnaround Time*, we simply use the Total Completion Time (*TCT*) for a job which is the sum of all these three *Times*. After send out signal that the server is free then the summation of Data Transfer Time (*DT*) and Queue Time (*QT*). It means that the server would not be working during *DT* or some time we assume that the server has other small tasks that can be interrupted when *DT* is complete. On the base of Scheduler Architecture Figure 3.7 and Scheduling Algorithm for TCT (3.2.3), the Total Completion Time is the summation of Queue Time (*QT*), Execution Time (*ET*) and Data Transfer Time (*DT*) i.e.

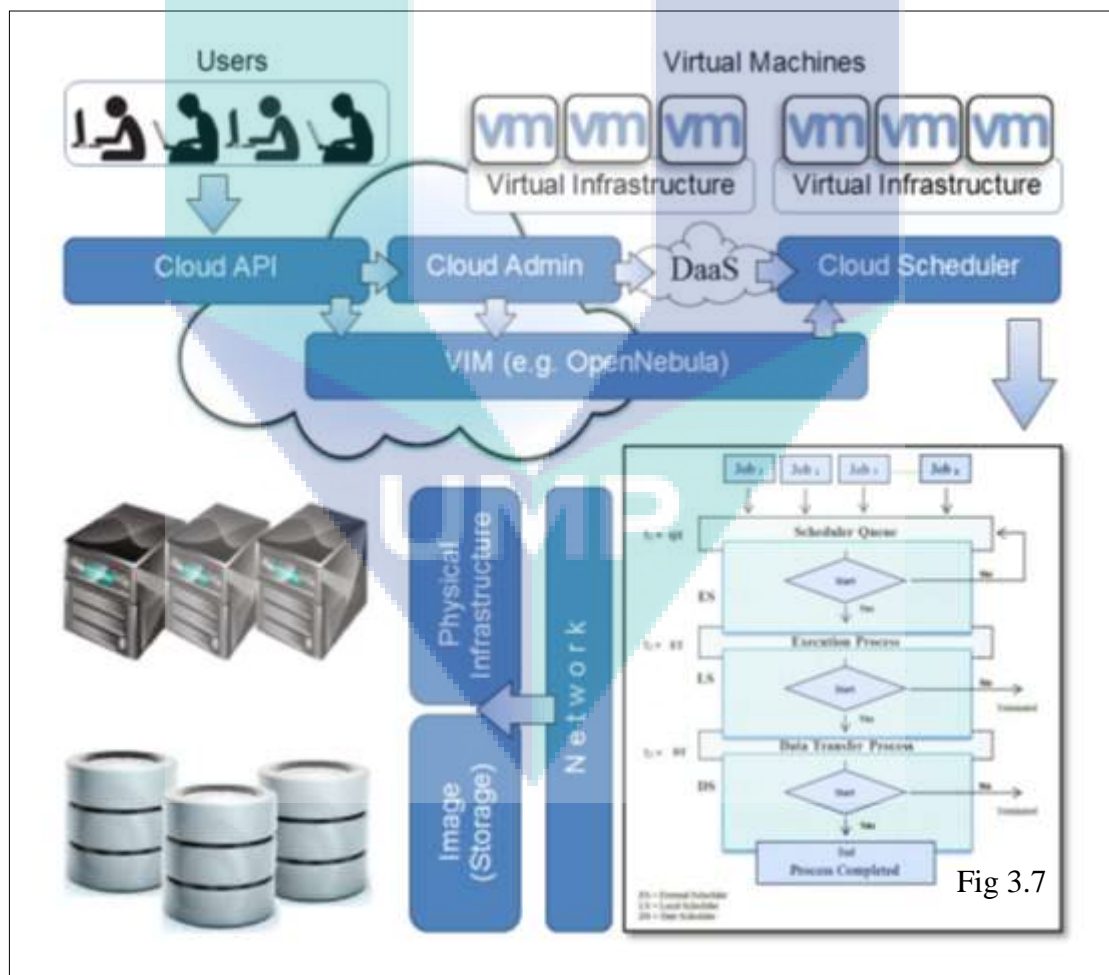
$$TCT_{j,i} = QT_{(i)} + ET_{j,i} + DT(f(j)) \quad (3.6)$$

It is possible that all jobs for that server are finished by the time Execution Time (*ET*) that the transfer finishes so the time to enter service is merely the Data Transfer time (*DT*). It is also possible that the some jobs are not finished when the data transfer completes so that the new job must wait until all previous jobs are completed. That time is just the system time (*ET*) of the current jobs at the server. Thus the total time that the

new job must wait before service is the Queue Time ( $QT$ ) i.e. system time ( $ET$ ) for all jobs currently at that server. That is why above architectures are taking maximum of  $QT$  and  $DT$  during the transfer time calculation with the addition of  $ET$  (Equation 3.5).

### 3.2.5 PHYSICAL STRUCTURE OF CLOUD ARCHITECTURE

A well-designed network of cloud architecture constructs a cloud-friendly network. To provide users with the same features found in commercial public clouds, private/hybrid cloud, software has some aspects on the base of which Cloud architecture has the following main components. As described in Figure 3.8, this section discusses all these components one by one.



**Figure 3.8:** Physical Structure of Cloud Architecture

### 3.2.5.1. *Application Programming Interface.*

Cloud Application Programming Interface (APIs) are used to build applications in the Cloud Computing market. Cloud APIs allow software to request data and computations from one or more services through a direct or indirect interface. Cross-platform interfaces have the advantage of allowing applications to access services from multiple providers without rewriting, but may have less functionality or other limitations versus vendor-specific solutions. An application-programming interface (API) is a key part of any web-based technology. Without APIs, the technology in question would serve a limited purpose and be severely confined in its capabilities. As Cloud Computing continues to grow and expand, APIs will also evolve to give developers an even more precise way of interaction. Although an essential piece, APIs in the cloud don't come without controversy.

### 3.2.5.2. *Cloud Administration*

Cloud administrators configure and maintain the cloud platform itself. User access, system software lifecycle, data center policy compliance are all necessary for a cloud in the same way that they are necessary for individual physical machines. However, cloud administration also differs from more traditional system administration in a few important ways. First, the cloud administrator must maintain two separate system software domains:

- System software on the physical machines.
- System software that is the platform itself.

It is often tempting to think of the cloud platform as being the 'operating system for the cloud'. This simplification, however, can lead to confusion particularly with respect to cloud installation and upgrade. An operating system 'boots' when a machine is started. A cloud is deployed and its components securely registered regardless of the running status of any given machine. Thus a cloud administrator must differentiate between operating system administration on the physical machines and administration

of the cloud platform itself which is a distributed and, in some sense, more abstract and unifying ‘machine’. Figure 3.8 describes the physical structure of cloud architecture.

### 3.2.5.3. *Data-as-a-Services.*

Data-as-a-Service (*DaaS*) is an information provision and distribution model in which data files (including text, images, sounds, and videos) are made available to customers over a network, typically the Internet (Search Cloud, 2013). *DaaS* offers convenient and cost-effective solutions for customer and client oriented enterprises. *DaaS* is emerging as underlying technologies that support Web services and SOA (*Service-Oriented Architecture*) mature. The evolution of SOA has greatly reduced the relevance of the particular platform on which data resides. *DaaS* adoption may be hampered by concerns about security, privacy, and proprietary issues. *DaaS* is totally concern with data transferring, hence this study and new proposed technique totally related with *DaaS*. Active provision of data depends on the correct time calculation for data transferring process. New proposed technique is concern with *DaaS*.

### 3.2.5.4. *Cloud Schedulers*

Cloud Scheduler acts after jobs reaching to queue. It looks at the job queue to discover which VM images are needed to complete the jobs in the queue, boots some VM images on the clusters it has access to. These VM images run the jobs from the queue, and Cloud Scheduler then shuts them down when they are no longer necessary. Cloud Scheduler further has three main components as discussed in section 3.2.2 and Figure 3.7 has described them. Besides, Cloud Scheduler performs the following responsibilities.

- Manage a queue of VMs
- Responsible for all aspects of a VM’s life-cycle
- Can be highly available, active-passive sets
- Manage thousands of execute nodes, and tens of thousands of active VMs, hundreds of thousands of pending VMs



- Other deployments scale further

#### 3.2.5.5. *Virtual Infrastructure Management (VIM)*

Management solutions for virtual infrastructure are a prime topic during virtualization discussions. As organizations deploy virtual solutions in more varied ways, new opportunities and challenges related to effective virtualization ecosystem management emerge. To provide users with the same features found in commercial public clouds, private/hybrid cloud software has the following aspects.

- Provide a uniform and homogeneous view of virtualized resources.
- Manage a VM's full life cycle, including setting up networks dynamically for groups of VMs and managing their storage requirements.
- Support configurable resource allocation policies to meet the organization's specific goals (high availability, server consolidation and so on).
- Adapt to an organization's changing resource needs, including peaks in which local resources are insufficient, and changing resources, including addition or failure of physical resources.

#### 3.2.5.6. *Cloud Networks*

A well-designed network of cloud architecture constructs a cloud-friendly network. IT should pursue an end-to-end approach to its network architecture, beginning with the user experience and the devices supported. The architecture also brings appropriate local area and wide area networking technologies and even multimedia optimization. By taking a holistic approach to networking, IT can lay the critical foundation to seamlessly rolling out cloud and on-premise services that accelerate the data availability and business innovation.

Cloud Schedulers Section 3.2.5.3 works on the base of queuing system. The following, Section 3.2.6 discusses queuing system with detail.

### 3.2.6 QUEUING SYSTEMS

In the queuing systems the queuing length process is the same to a birth-and-death process, a linearly structured *Markovian* process with only one-step change. Additional subjects include the departure processes and busy periods. These outcomes are applied, in Kendall's classification by (Stallings, 2000; and Blanc, 2011). i.e. for the single server  $M/M/1$  system, multi-server  $M/M/c$  system, finite queue-capacity  $M/M/c/K$  system, variants of the foregoing systems with *balking* and *reneging* jobs and with state-dependent service rates. But here this study is using finite jobs-population  $M/M/c/K/P$  system because of the need to know the size of jobs prior starting the transfer of data by using new architecture. (Stallings, 2000; and Blanc, 2011) have discussed these models and further with great details by (Lipsky, 2009; and Jain et al., 2007). This study has used the  $M/M/c/*P$  (Finite Jobs Population Model).

#### 3.2.6.1 $M/M/c/*P$ (Finite jobs population)

The  $M/M/c/K/P$  queuing system is another variant of the  $M/M/c$  queuing system. In this system, there is a finite number of potential jobs  $P$  (General notation for population is  $M$ ), while there is room (queue) for jobs  $K$ , including the jobs in service  $P \geq K \geq C$ ). This is the fact that a closed system where each of the  $P$  job is either inside the system (waiting or being served) or passive outside the system until its next turn to visit the system. In this case, the arrival rate depends on the number of passive jobs so that the arrival process is not a Poisson process. It is assumed that each potential job returns to the system after an exponentially distributed passive time with rate  $\lambda$ . Such a system is stable for every positive value of  $\lambda$ , and service rate  $\mu$ . Special case is the  $M/M/c/c/P$  system which is normally referred to as Engset loss system (Stallings, 2000; Blanc, 2011; and Lipsky, 2009).

If there are  $n$  jobs in the queue, then there are  $N-n$  jobs in the source. Assuming that jobs waiting in the source are exponentially distributed amount of time with the average before returning to the queue, and that they are independent of each other. If

there are  $N-n$  jobs in the source, then they create a Poisson input process to the queue with rate  $\lambda$ .

Finite Population Model (Lipsky, 2009; and Jain et al., 2007) have shown as below:

$$\lambda_n = \begin{cases} (M-n)\lambda & \text{for } 0 \leq n < M \\ 0 & \text{for } n \geq K \end{cases}$$

Here  $M$  represents the total size of the population which is the total number of jobs. Assuming a system with  $C < M$  service units (Stallings, 2000; Blanc, 2011; and Lipsky, 2009) i.e.

$$\mu_n = \begin{cases} n\mu & \text{for } 1 \leq n < c \\ c\mu & \text{for } n \geq c \end{cases} \quad (3.7)$$

To calculate  $L_q$  and wait in queue  $W_q$  or  $QT$ , we need probability of '0' entity in the system  $P_0$ , probability of 'n' entities in the system  $P_n$ , average number of jobs in the system  $L$ , and average time spent in the system  $Ws$  ( $ET$ ).

Hence,

$$P_n = \begin{cases} \left( \frac{M!}{(M-n)!n!} \right) \rho^n P_0 & \text{for } 0 \leq n < c \\ \left( \frac{M!}{(M-n)!n!} \right) \frac{n!}{(c^{n-c}c!)} \rho^n P_0 & \text{for } c \leq n \leq M \end{cases} \quad (3.8)$$

Where,

$$\binom{M}{n} = \frac{M!}{(M-n)!n!},$$

To express the binomial coefficient, apply  $\sum_{n=0}^M P_n = 1$  and solve for  $P_0$

$$p_0 = \left[ \sum_{n=0}^{c-1} \left( \frac{M!}{(M-n)!n!} \right) \rho^n + \sum_{n=c}^M \left( \frac{M!}{(M-n)!n!} \right) \frac{n!}{c^{n-c}c!} \rho^n \right]^{-1} \quad (3.9)$$

For easy understanding, Figure 3.9 explains some parameters. Where  $\rho$  is the utilization; the fraction of time facility (server or servers) is busy.  $N(t)$  is the number of jobs in the queuing system at time  $t$ ,  $A_i$  is the Arrival time, and  $D_i$  is the Departure time,  $S_i$  is the Service (consumed) Time for a job in the queuing system.

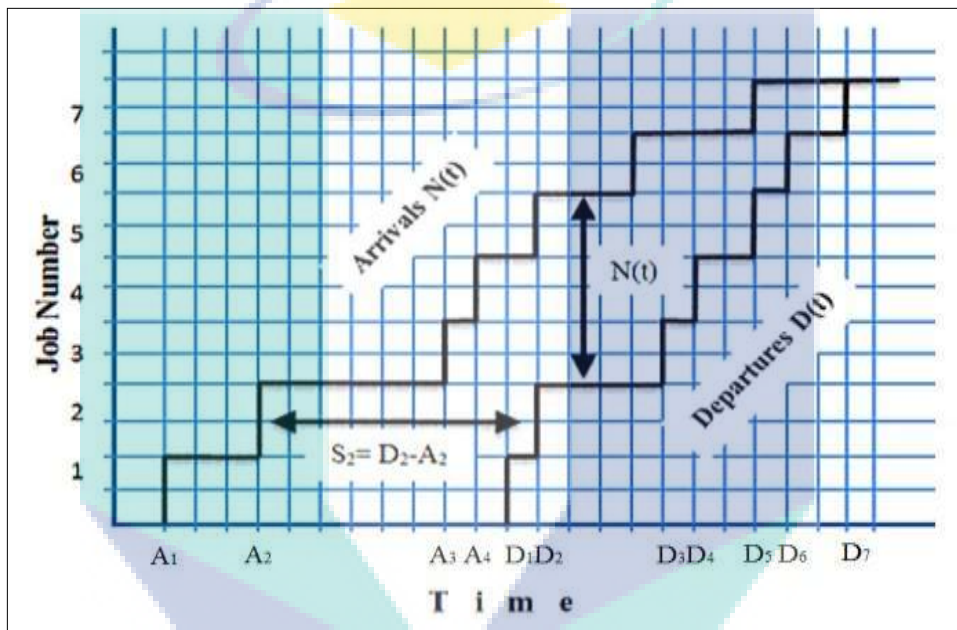


Figure 3.9: Arrivals and Departures from a system

Using the above formulas, after the calculation  $P_0$  &  $P_n$ , it is easy now to derive the average size of the system, which is the number of jobs in the system.

$$L = \left[ \sum_{n=0}^{c-1} n \left( \frac{M!}{(M-n)!n!} \right) \rho^n + \sum_{n=c}^M n \left( \frac{M!}{(M-n)!n!} \right) \frac{n!}{c^{n-c}c!} \rho^n \right] p_0 \quad (3.10)$$

$$L = \sum_{n=c}^M (n-c)p_n = \sum_{n=c}^M np_n - c \sum_{n=c}^M p_n$$

$$= L - c + p_0 \sum_{n=0}^{c-1} (c-n) \left( \frac{M!}{(M-n)!n!} \right) \rho^n \quad (3.11)$$

In order to describe the *waiting time* and *service time* by using Little's Law, we need to know, the means arrival rate of the jobs. With  $n$  jobs already in the system, a maximum of  $M-n$  jobs remains outside the arrival time. Its outcomes in a mean average arrival rate of  $(M-n)\lambda$ , which is shown below:

$$\begin{aligned} \bar{\lambda} &= \sum_{n=0}^M (M-n)\lambda p_n = \lambda \left( M \sum_{n=0}^M p_n - \sum_{n=c}^M n p_n \right) \\ &= \lambda(M-L) \end{aligned} \quad (3.12)$$

By using Little's Law (Stallings, 2000; Blanc, 2011; and Lipsky, 2009), the mean arrival rate  $\lambda$  can produce  $W_s$  and  $W_q$ , which is shown below:

$$W_s = ET = \frac{L}{\lambda(M-L)} \quad (3.13)$$

and

$$W_q = QT = \frac{L_q}{\lambda(M-L)} \quad (3.14)$$

Consuming time in the execution process by a processor is given below.

$$W_s = ET(j) = L - c + p_0 \sum_{n=0}^{c-1} (c-n) \left( \frac{M!}{(M-n)!n!} \right) \rho^n \quad (3.15)$$

Now the Data Transfer Time, according to (Nguyen and Lim, 2007) is:

$$T_t = DT(f(j)) = \text{size}(k) / BW_{(i,j)} \quad (3.16)$$

$\text{Size}(k)$  is the *File Size* in bytes, and  $BW$  is the available used *bandwidth* between computing sites.

### 3.2.7 PROPOSED TECHNIQUE

Now the values of Equations 3.14, 3.15, and 3.16, are:

$$QT = \frac{L_q}{\lambda(M-L)}$$

$$ET(j) = L - c + p_0 \sum_{n=0}^{c-1} (c-n) \left( \frac{M!}{(M-n)!n!} \right) \rho^n$$

$$DT(f(j)) = \text{size}(k) / BW_{(i,j)}$$

Putting  $QT$ ,  $ET$  and  $DT$  in Equation 3.6, result is new proposed technique:

$$TCT_{ji} = QT_{(i)} + ET_{j, i} + DT(f(j))$$

$$TCT_{ji} = \frac{L_q}{\lambda(M-L)} + L - c + p_0 \sum_{n=0}^{c-1} (c-n) \left( \frac{M!}{(M-n)!n!} \right) \rho^n + \text{Size}(k) / BW_{(i,j)} \quad (3.17)$$

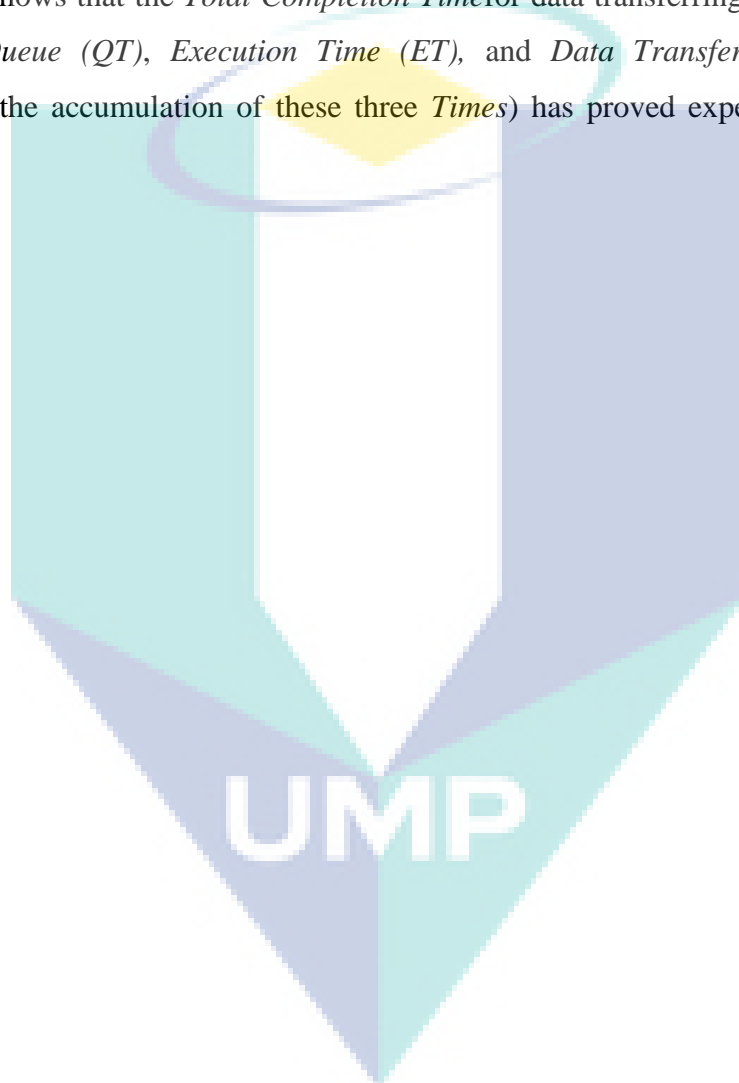
### 3.3 SUMMARY

Dealing with the large amount of data makes the requirement more critical for accuracy and efficiency during the access of data. This study has introduced new scheduling techniques to support the calculation process for the *Total Completion Time (TCT)* for the transfer of data. In order to obtain *TCT*, we need to calculate *Queue Time (QT)*, *Execution Time (ET)*, and *Data Transfer Time (DT)* for a job.

Previous works (Nguyen and Lim, 2007; Tang et al., 2006; and Ranganathan and Foster, 2003) are taking maximum of *QT* and *DT*. New proposed scheduling technique *TCT* gives importance to each parameter by considering each one separately, while calculating the *Total Time of Completion*. We have to add all these parameters because

all of them are different; and each one has its own importance and no one is insignificant to be ignored. In addition, for the evaluation of the new technique, this study has used  $M/M/C/*/P$  queuing models.

There is a great impact on accuracy by taking each parameter separately in the formula for the *Total Completion Time (TCT)* during job transferring process. Hence this study shows that the *Total Completion Time* for data transferring can be the sum of *Time in Queue (QT)*, *Execution Time (ET)*, and *Data Transfer time (DT)*. New technique (the accumulation of these three *Times*) has proved experimentally in next chapter.



## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 INTRODUCTION

In this chapter, the implementation of the new proposed model *Total Completion Time (TCT)* will be explained. The purpose of this implementation is to illustrate the given model for *TCT* in previous chapter and to show that *TCT* can be used in practical application. To bring *TCT* in practical shape by using all mentioned Equations in previous sections, simple user interface has been used. With its help, this study compares the current results of *TCT* with *Total Transfer Time (TT)* model of (Nguyen and Lim, 2007; and Tang et al., 2006).

As discussed in Chapter 3, the scheduling architecture is encapsulated in three distinct modules, Figure 3.7, on the basis of scheduler's three main portions, this study also divides the *Total Transfer Time* for data from one node to another in three *Times*. First, *QT (Queue Time)* is the time a job passes in the queue, starting immediately after entrance to the queue and before starting the execution. Second, the time denoted by *ET (Execution Time)* is the time after entrance from the queue and before starting the transfer process. Third, *DT* is the *Data Transfer Time*, starts after execution completion and before starting data transferring process. All these *times* are different from each other and each one has its own importance. Hence, it means that the *Total Completion Time (TCT)* for a job is the sum of all these three times, i.e. *Queue Time (QT)*, *Execution Time (ET)*, and *Data Transfer Time (DT)*. As a result, by treating all these parameters separately, there is a very beneficial change to improve data efficiency and reliability (in terms of accuracy in calculating the transfer time).



## 4.2 Parameters Used

The used of parameters in experiments are on the basis of definition in chapter 3, Section 3.1.1, page 41. This section has divided all parameters in two Dependent and Independent categories. For overall comparison Finite Population Model has used.

## 4.3 M/M/c\*/P Model

In order to evaluate the performance of proposed architecture, we have used the queuing calculators (Queuing Theory Calculator, 2012) for calculations and comparison of results with already used formulae (Nguyen and Lim, 2007; and Tang et al., 2006). In queuing systems, there are four models, namely a) single server  $M/M/1$  system, b) multi-server  $M/M/c$  system, c) finite queue-capacity  $M/M/c/K$  system, and d) finite jobs-population  $M/M/c/K/P$  system. Here we used  $M/M/c/K/P$  because for Transfer Time Calculation, we need to know the total population  $M$ , which is the number of jobs. For  $TCT$ , we need to calculate the values for all parameters mentioned in Section 3.1.1, with the help of formulae used by (Queuing Theory, 2011; Intro to Queuing Theory-Birth Death processes, 2011; Lec-30, 2010; Lec-31, 2010; and Lec-32, 2010). Equations described above Equation 3.9, Equation 3.11, Equation 3.15, Equation 3.15, and Equation 3.16 simultaneously.

$$p_0 = \left[ \sum_{n=0}^{c-1} \left( \frac{M!}{(M-n)!n!} \right) \rho^n + \sum_{n=c}^M \left( \frac{M!}{(M-n)!n!} \right) \frac{n!}{c^{n-c}c!} \rho^n \right]^{-1}$$

$$L = \left[ \sum_{n=0}^{c-1} n \left( \frac{M!}{(M-n)!n!} \right) \rho^n + \sum_{n=c}^M n \left( \frac{M!}{(M-n)!n!} \right) \frac{n!}{c^{n-c}c!} \rho^n \right] p_0$$

$$L_q = L - c + p_0 \sum_{n=0}^{c-1} (c-n) \left( \frac{M!}{(M-n)!n!} \right) \rho^n$$

$$W_s = ET = \frac{L}{\lambda(M-L)}$$

$$W_q = QT = \frac{L_q}{\lambda(M-L)}$$

#### 4.4 Queuing Theory Calculator

In previous Chapter, each parameter of queuing theory has defined. Now it is easy to use the discussed formulae in Section 4.3, and calculate the value of  $TCT$  to obtain results according to the new propose technique. For these calculations, in first phase MS Excel, and then Queuing Theory Calculator interface (Queuing Theory Calculator, 2012), Figure 4.1 has used. In this interface, any queuing model can be used, but for the evaluation of the new proposed model, the finite population model is using. The Finite population model is denoted by  $M/M/C/*/M$  or  $M/M/c/*/P$ .

**Queueing theory calculator**

Choose queuing model:  MMC,  MMInf,  MMCK,  MMC\*/M

Space for calculations: Eg. insert 2+2 then press Res.

Data analysis: Insert list. Press Analyze to get goodness of fit p-value.

Res Analyze

C (No. of Servers)	K (Queue capacity)	M (Entities population)	λ (incoming rate)	μ (service rate)	24 hrs/day
1	NaN	4	1 units in/Sec	1 units out/Sec	Calculate Clear Form

Round to 4 decimal places.

ρ (Server utilization)	L (Average entries in system)	Lq (Average entries in queue)	W (secs) (Average time spent in system)	Wq (secs) (Average time waiting in line)	P0	Pn (Probability of n entities being in the system)
0.0154	3.0154	2.0308	3.0625	2.0625	0.9846	0.0154

**Figure 4.1:** Queuing Theory Calculator

Steps for using the Queuing Theory Calculator

1. Choose the queuing model,  $M/M/C$  (or  $M/M/1$  if  $C=1$ ),  $M/M/Inf$ ,  $M/M/C/K$ , or  $M/M/C/*/M$ .
2. Input the number of servers in the system ( $C$ ).
3. Give queue capacity ( $K$ ), the maximum number of entities that the queue can hold ( $K$ ). For  $M/M/C/*/P$  Model, no involvement of  $K$  and no item/s waiting outside the queue. Hence no need to put the value of  $K$ .

4. Mention the entire population ( $M$ ) of the jobs, the maximum number of existing entities that need to be processed.
5. Choose the incoming jobs arrival rate Lambda ( $\lambda$ ), which is the number of jobs coming per unit time.
6. Provide service rates Mu ( $\mu$ ), which is the number of job services given per unit time.
7. After steps 5 and 6, there is an option for units, in practice sometimes get the incoming service rates with different units.
8. Press Calculate.
9. Values will come for server utilisation Ro ( $\rho$ ), Average entities in the whole system ( $L$ ), Average entities in queue ( $L_q$ ), Average time of an entity spends in the system ( $W$ , for easy understanding we use  $W_s$  which is ET *Execution Time*), Average time of an entity waiting in line to be served ( $W_q = QT$ ), the probability that there would be exactly ' $n$ ' entities in the system at a certain point ( $P_n$ ).
10. The value of ' $n$ ' can be modified as desired, the probability that an entity will spend in line exactly or less than ' $n$ ' units of time ( $T_q$ ) and the probability that an entity will spend exactly or less than ' $n$ ' units of time in the system ( $T$ ).
11. For quick calculations we can use the given 'space for calculations' by inputting the desired formula and then press the 'Res' button, e.g. input '2+2' then press [Res]. Number four will be displayed.

**Example:** Suppose we want to calculate  $W_s = ET$ ,  $W_q = QT$ ; the number of jobs means population ( $M$ ), which is 4, for which we need to calculate the transfer time. If  $C = 1$ ,  $M = 4$ ,  $\lambda = 1$ ,  $\mu = 1$  then we can use the above formulas 3.11, 3.13, 3.15, 3.16 to calculate  $L$ ,  $ET$ ,  $QT$ ,  $W$  and  $DT$  respectively, assuming that 1 KB = 1 job, and  $n = 1$ , where  $n$  is the job(s) number in the system.

$$P_0 = \left[ \sum_{n=0}^{c-1} \left( \frac{M!}{(M-n)!n!} \right) \rho^n + \sum_{n=c}^M \left( \frac{M!}{(M-n)!n!} \right) \frac{n!}{c^{n-c}c!} \rho^n \right]^{-1}$$

$$p_0 = \left[ \sum_{n=0}^{1-1} \left( \frac{4!}{(4-1)!1!} \right) \rho^1 + \sum_{i=1}^4 \left( \frac{4!}{(4-1)!1!} \right) \frac{1!}{1^{1-1}1!} \rho^1 \right]^{-1}$$

$$= 0.0154$$

$$L = \left[ \sum_{n=0}^{c-1} n \left( \frac{M!}{(M-n)!n!} \right) \rho^n + \sum_{n=c}^M n \left( \frac{M!}{(M-n)!n!} \right) \frac{n!}{c^{n-c}c!} \rho^n \right] p_0$$

$$= 3.0154$$

$$L_q = L - c + p_0 \sum_{n=0}^{c-1} (c-n) \left( \frac{M!}{(M-n)!n!} \right) \rho^n$$

$$= 3.0154 - 1 + 0.0154 + (1-1) \times [4] \times 1$$

$$= 2.0308$$

$$W_s = ET = \frac{L}{\lambda(M-L)} = \frac{3.0154}{1(4-3.0154)} = 3.0625$$

$$W_q = QT = \frac{L_q}{\lambda(M-L)} = \frac{2.0308}{1(4-3.0154)} = 2.0625$$

#### 4.5 Transfer Time Calculations

Using the interface below, we can get *Data Transfer Time (DT)* by applying a very simple formula, that is,  $DT = \text{File size} / \text{Bandwidth}$  used by (Nguyen and Lim, 2007; and Tang et al., 2006). Data Transfer Speed Calculator, from Figure 4.2 (T1 the complete telecom source, 2012) has been used to calculate the *Data Transfer Time (DT)*.

The image shows a web-based calculator titled "File Transfer Time - Data Transfer Speed Calculator". It has three main input areas: "File Size" with the value "18", "File Size Units" with a dropdown menu open showing "Kilobytes" selected, and "Select Transfer Speed" with a dropdown menu open showing "56 Kbps" selected. The "Resulting time" field shows "2.63 Seconds".

**Figure 4.2:** Transfer Time Calculator

Steps for using the Data Transfer Calculator

1. Input size for desired file which needs to be transferred, in the File Size textbox.
2. Select the file size unit from the List Box.
3. Select transfer speed (Bandwidth), through which file will transfer.

For *Transfer Time* calculation, it is extremely necessary and important to note that this time is only the transfer time, exclude the *Queue Time* and *Execution Time*. Additionally, this time starts immediately after finishing the execution process in a server until reaching the destination, i.e. till the completion of transfer process. During calculation, if the bandwidth unit is *bps* then we need to convert the file size unit to *bit*.

**Example 1:** Suppose we want to calculate the Transfer Time ( $T_t=DT$ ) for 18Kbytes data, by using 56Kbps bandwidth. For this purpose we need to use  $T_t=Filesize/Bandwidth$  formula. Here, the required unit is bit, which is easy to achieve from KB, and bps from Kbps. i.e. 18Kbytes =  $18 \times 1024 \times 8 = 147456$  bits and 56Kbps = 56000bps.

Substituting values in the Equation 3.16, it will give the following result.

$$Tt = DT = \text{File Size/Bandwidth} = 147456 / 56000 = \mathbf{2.6331} \text{ seconds}$$

**Example 2:** Suppose we need to transfer 5GB data through 1.544Mbps, calculation will be as follows:

$$5\text{GB} = 5 \times 1024 \times 1024 \times 1024 \times 8 = 42949672960 \text{ bits}$$

$$1.544\text{Mbps} = 1.544 \times 1000 \times 1000 = 1544000\text{bps}$$

$$Tt = DT = \text{File size/Bandwidth} = 42949672960 / 1544000 \\ = \mathbf{27817.145699} \text{ sec} = 7 \text{ Hours } 43 \text{ Minutes } 37.15 \text{ Seconds}$$

**Example 3:** Now we want to calculate  $Tt$  for 4KB through 56Kbps, for which we already have calculated  $Ws$  and  $Wq$ . Calculation is shown below:

$$4\text{KB} = 4 \times 1024 \times 8 = 32768 \text{ bits}$$

$$56\text{Kbps} = 56 \times 1000 = 56000\text{bps}$$

$$Tt = DT = \text{File size/Bandwidth} = 32768 / 56000 \\ = \mathbf{0.585142} \text{ sec}$$

Now in order to compare the results of the new proposed technique and existing technique by (Nguyen and Lim, 2007; and Tang et al., 2006), the first interface (Figure 4.2) calculates Queue Time ( $QT$ ), Execution Time ( $ET$ ), and the second interface (Figure 4.3) calculates the Data Transfer Time ( $DT$ ). By adding all these three parameter values, we can get the *Total Completion Time (TCT)*. Suppose the Transfer Time to be calculated for 4 KB by using 56Kbps. The values can be compared (in terms of accuracy). According to the existing formulas by (Tang et al., 2006),  $ET$  can be calculated as below: Where  $QT_{(i)} = Wq$ ,  $ET_{k,i} = Ws$  and  $DT(f(k),i) = Tt$

Putting Values in Equation 3.5,

$$TT_{k,i} = \max\{QT_{(i)}, DT(f(k),i)\} + ET_{k,i}$$

$$TT_{k,i} = \max(2.0625, 0.585) + 3.0625 = \mathbf{5.125}\text{sec}$$

According to the new proposed technique, Equation 3.6:

$$TCT_{ji} = QT_{(i)} + ET_{j,i} + DT(f(j))$$

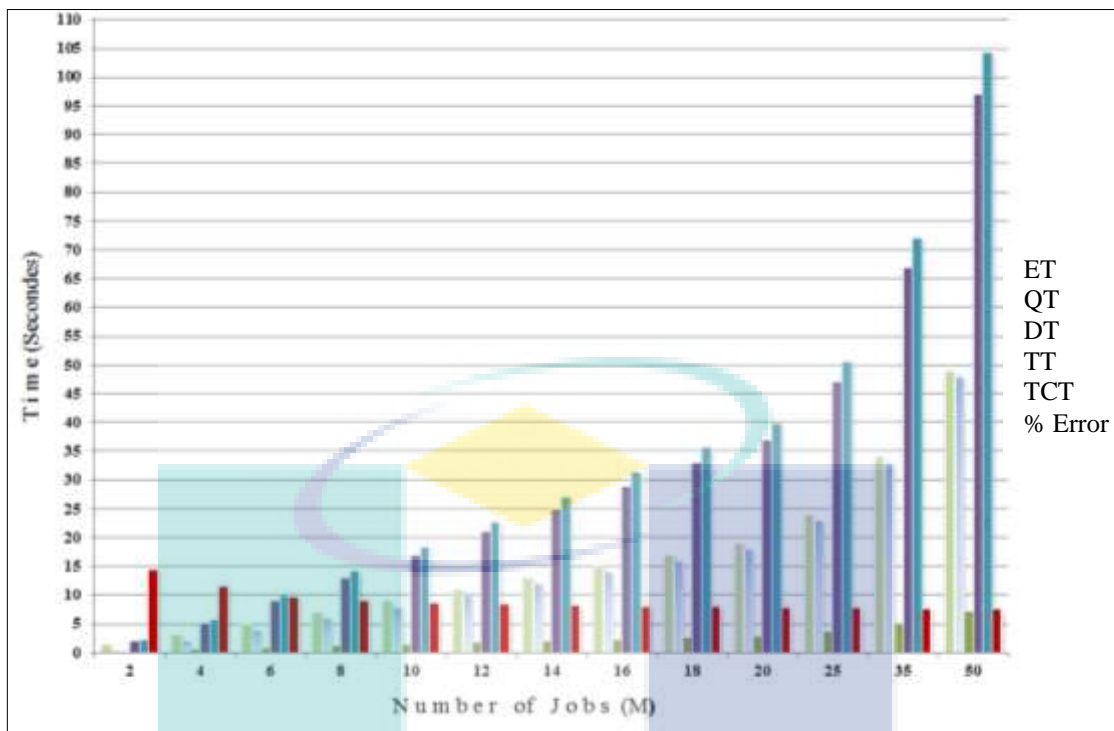
$$TCT_{ji} = 2.0625 + 0.5851 + 3.0625 = \mathbf{5.7101}\text{sec}$$

Here, by adding one extra ignored minimum parameter  $DT$ , which is equal to 0.5851sec, our result is little bit high when compared to the existing technique's result. However, by adding and giving importance to each parameter, which are  $QT$  and  $DT$ , it is very clear that both parameters are different from each other, and each one has own importance. No one parameter is ignorable, and both need to give them their importance. Hence, this new technique shows more efficiency (in terms of accuracy) than the existing models. As a result, the *Total Completion Time (TCT)* can be the sum of all three parameters which are, *Queue Time (QT)*, *Execution Time (ET)*, and *Data Transfer Time (DT)*.

#### 4.6 Analysis and Results

To evaluate a new model, by using the discussed formulae in Chapter 3, first of all we have to calculate the total consume time in data *transferring* process. *Transfer Time (DT)* excludes the *Queue Time (QT)* and *Execution Time (Ws)*. By using the new mathematical technique, we calculate the difference between existing Equations 1.1 and new Equation 3.17. Twenty experiments have been conducted for analysis. Changes occur on the basis of variation in population ( $M$ ) which is the number of jobs, Bandwidth ( $BW$ ) and number of servers ( $C$ ). Other independent parameters have been kept constant, i.e.  $\lambda$  and  $\mu$ .

The results below depicted the total *Transfer Time (TT)* calculated by using Equation 3.5, and *Total Completion Time (TCT)* calculated by Equation 3.17. Figure 4.3 shows the comparison between  $TT$  and  $TCT$ .



**Figure 4.3:** General comparison of TT and TCT

The number of jobs or population, arrival rate of the jobs, service rate, number of servers, number of jobs in queue, number of jobs in the system, mean waiting time in queue, mean waiting time in system, bandwidth, data transfer time are denoted by  $M$ ,  $\lambda$ ,  $\mu$ ,  $c$ ,  $Lq$ ,  $L$ ,  $QT$ ,  $ET$ ,  $BW$ , and  $DT$  respectively. The Total Transfer Time, according to the existing formula, is denoted by  $TT$  and the Total Completion Time according to the new proposed formula is denoted by  $TCT$ . Table 4.1 compares TT and TCT using the above-mentioned notation.

In order to calculate the Total Completion Time for transferring data, first we need to calculate wait in the system ( $ET$ ), wait in queue ( $QT$ ), and transfer time ( $DT$ ). After getting the values of these three parameters, results comparison can be conducted between existing and new techniques with different aspects.

The parameters in the existing and new techniques are  $QT_{(i)} = Wq$ ,  $ET_{k,i} = Ws$  and  $DT(f(k),i) = Tt$ . For  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 512Kbps$ . By using



Equations 3.11, 3.13, 3.14 and 3.15, first we calculate  $L$ ,  $W$ ,  $QT$ ,  $ET$  respectively and then by using Equation 3.17, we calculate  $TCT$ . The comparison of the results of Equation 3.5 and Equation 3.6 describes that new propose technique for  $TCT$  shows more accuracy, by adding the ignored parameter in Equation 3.6 is either  $Wq$  or  $DT$ . It means that the Total Time for Transferring Data can be the sum of *Time in Queue*, *Execution Time*, and *Data Transfer Time*.

**Table 4.1:** Comparison of various aspects for queuing process

Population	Wait in Queue	Wait in System	Transfer Time	Total Completion Time Using Existing Formula	Total Completion Time Using New Propose Formula	Error Estimation % error = $\frac{(\text{estimate} - \text{actual})}{\text{actual}} * 100$	% Accuracy = $100 - \% \text{ Error}$
M	QT	ET	DT	TT	TCT	% Error	% Accuracy
2	0.5000	1.5000	0.0300	2.0000	2.0300	1.5000	98.5000
4	2.0625	3.0625	0.0600	5.1250	5.1850	1.1707	98.8293
6	4.0031	5.0031	0.1000	9.0062	9.1062	1.1103	98.8897
8	6.0001	7.0001	0.1300	13.0002	13.1302	1.0000	99.0000
10	8.0000	9.0000	0.1600	17.0000	17.1600	0.9412	99.0588
12	10.0000	11.0000	0.1900	21.0000	21.1900	0.9048	99.0952
14	12.0000	13.0000	0.2200	25.0000	25.2200	0.8800	99.1200
16	14.0000	15.0000	0.2600	29.0000	29.2600	0.8966	99.1034
18	16.0000	17.0000	0.2900	33.0000	33.2900	0.8788	99.1212
20	18.0000	19.0000	0.3200	37.0000	37.3200	0.8649	99.1351
25	23.0000	24.0000	0.4000	47.0000	47.4000	0.8511	99.1489
35	33.0000	34.0000	0.5600	67.0000	67.5600	0.8358	99.1642
50	48.0000	49.0000	0.8000	97.0000	97.8000	0.8247	99.1753

#### 4.7 Variation of TT and TCT with bandwidth change

This experiment compares the difference between both times ( $TT$  and  $TCT$ ) with the change of bandwidth. Other parameters are the same, only bandwidth is changing. For  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 56Kbps$ , Figure 4.4 shows the comparison between  $TT$  and  $TCT$  by using  $BW$  56Kbps. Results show the changes in  $TCT$  with population ( $M$ ) variation. With change of bandwidth only  $DT$  will change, because there is no involvement of bandwidth during the data processing stage in the queue as well as in the system.

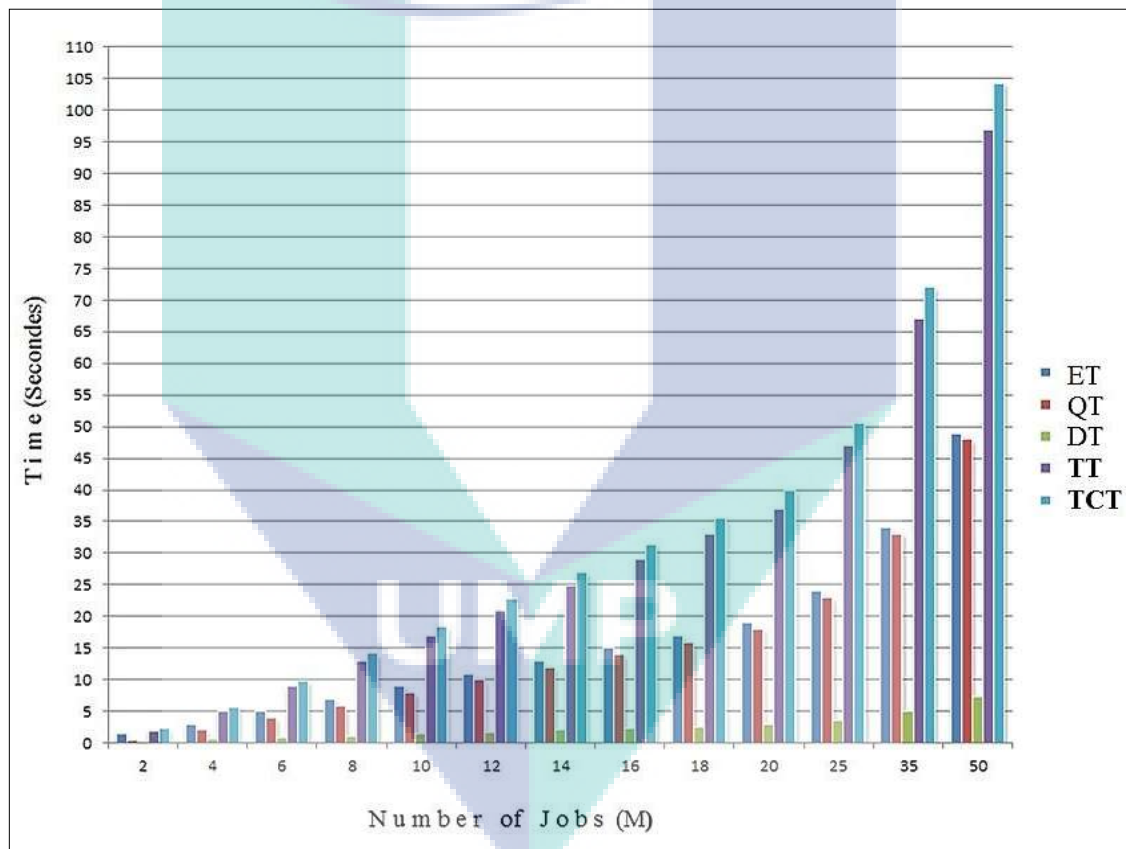


Figure 4.4: TT and TCT with 56Kbps

#### 4.7.1 Bandwidth variation effect

In this experiment, bandwidth ( $BW$ ) has changed; other parameters are the same. For  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 128Kbps$ . Figure 4.5 shows comparison between  $TT$  and  $TCT$  by using  $BW$  128Kbps. Results indicate the changes with the variation of population  $M$ . Bandwidth variation will change only  $DT$ .

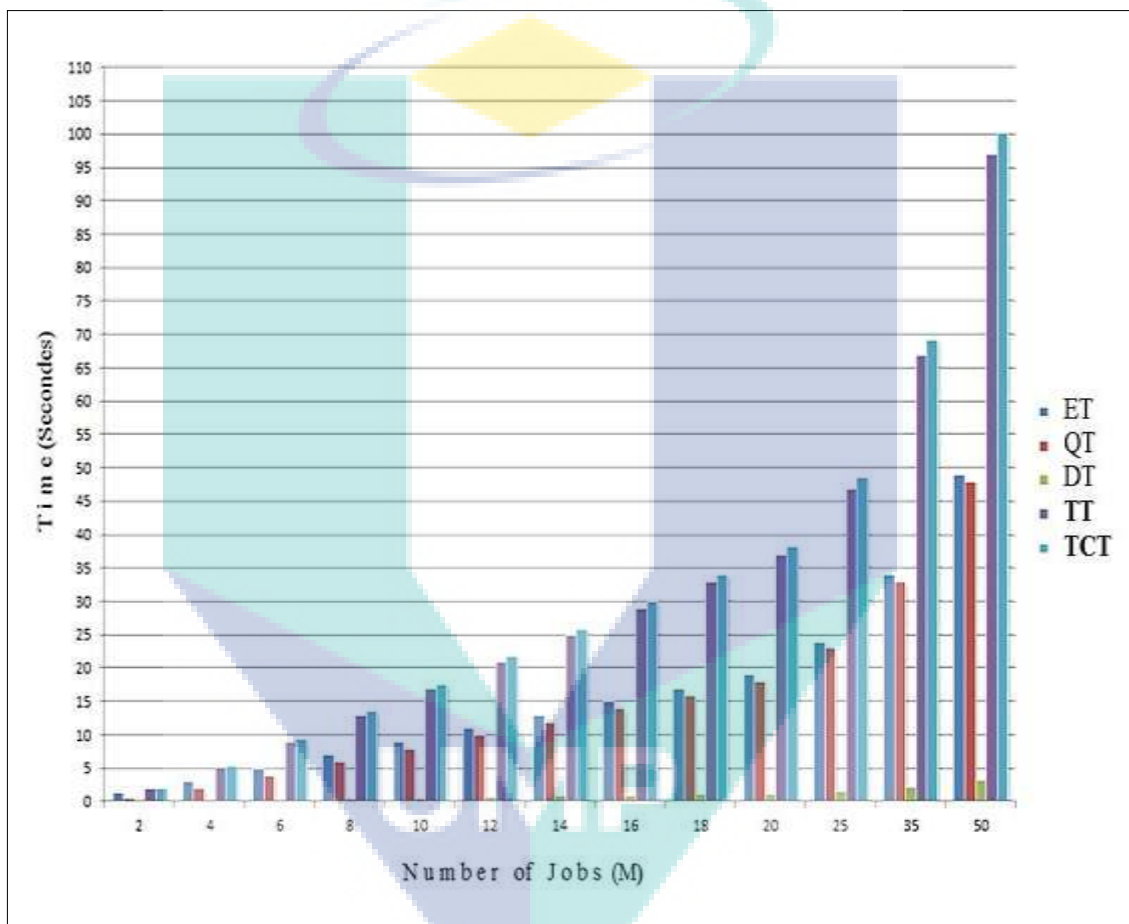
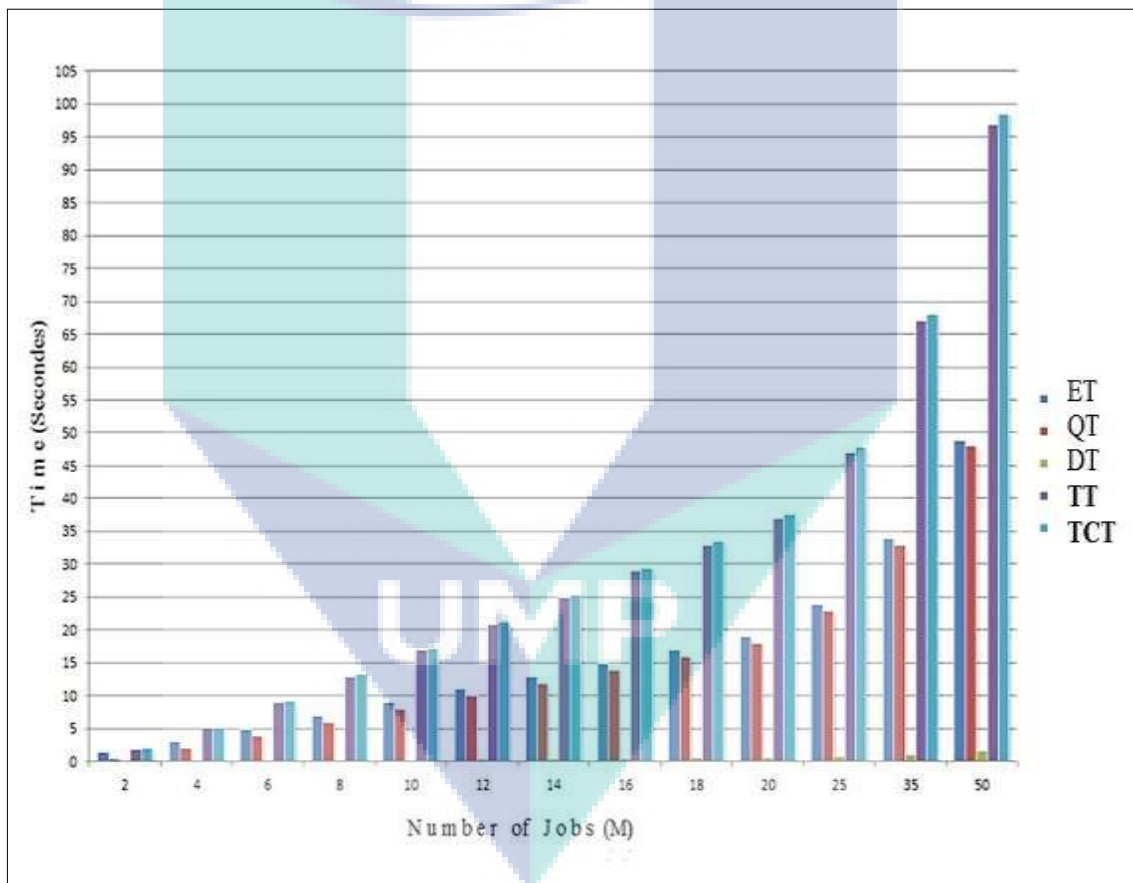


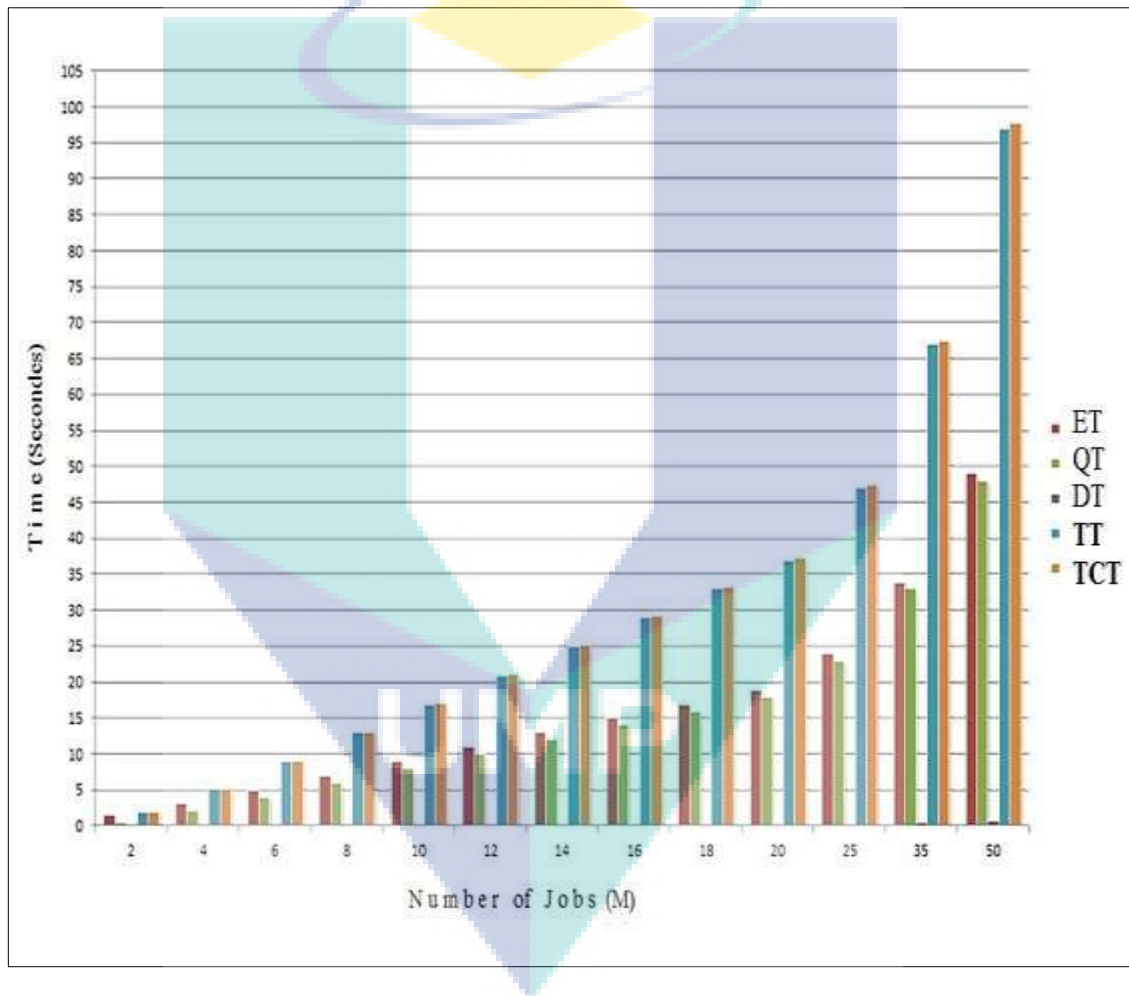
Figure 4.5: TT and TCT with 128Kbps BW

This experiment describes the comparison for both times ( $TT$  and  $TCT$ ). The objective is to know the variation with change of bandwidth  $BW$  from 128 to 256Kbps. Other parameters have the same value, for  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 256Kbps$ . Figure 4.6 shows the comparison between  $TT$  and  $TCT$ . The results show the gap between both times, i.e.  $TT$  and  $TCT$ . Gap is decreasing by using high bandwidth. These results can be compared with previous Figure 4.5 and next coming Figure 4.7, where 128 and 512Kbps bandwidth is used respectively. Change comes only in  $DT$  with the change in bandwidth. As mentioned above, there is no involvement of bandwidth during the data processing stage in the queue as well as in the system.



**Figure 4.6:** TT and TCT with 256Kbps BW

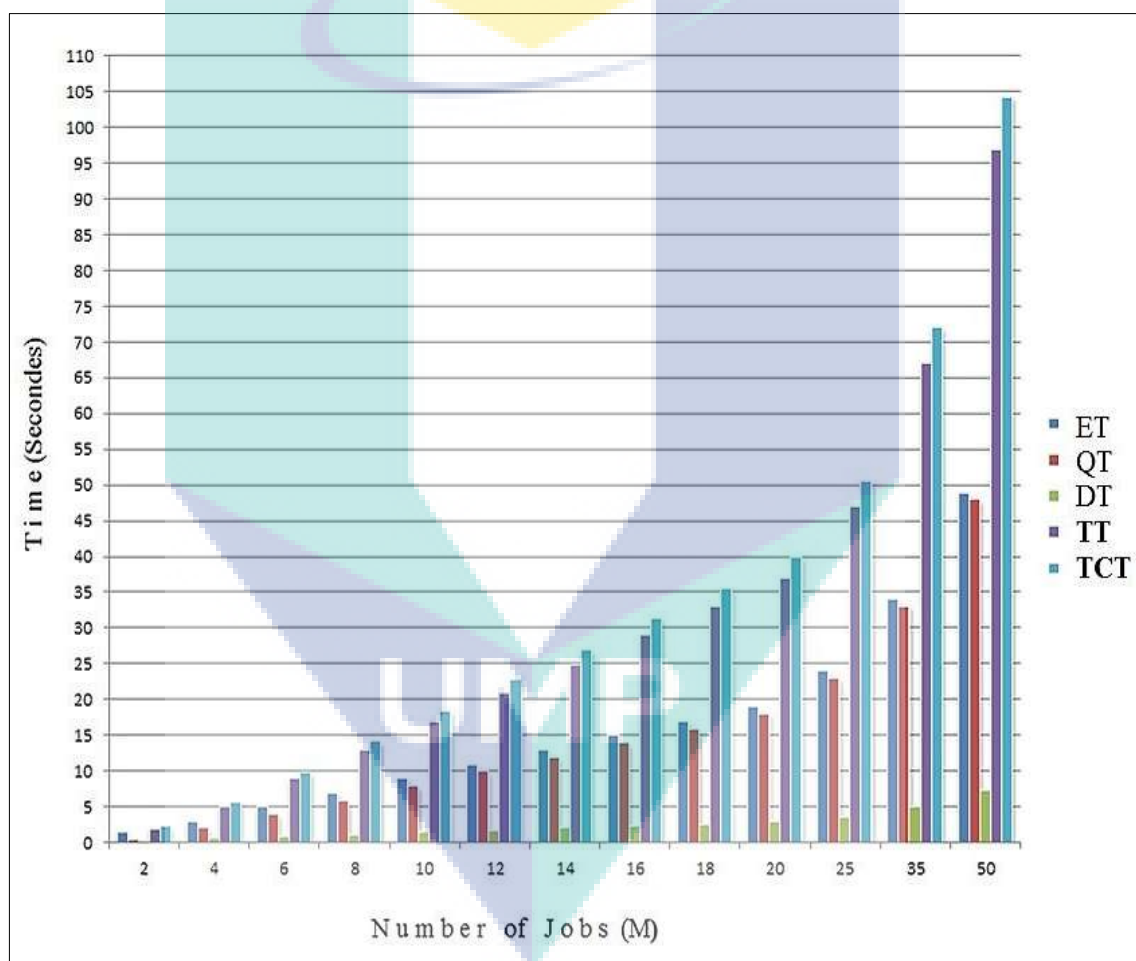
Figure 4.7 shows the comparison between  $TT$  and  $TCT$  by using  $BW$  512Kbps. These results can be compared with previous Figure 4.6, where 256Kbps bandwidth has used. After comparing all these Figures, Figure 4.4, Figure 4.5, Figure 4.6 and Figure 4.7, the difference between  $TT$  and  $TCT$  has decreased continuously. It means that the new proposed model is more efficient for large data transferring. For Figure 4.5,  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 512Kbps$ . Figure 4.5 shows the changes in  $TT$  and  $TCT$  due to the change in  $DT$ .



**Figure 4.7:** TT and TCT with 512Kbps BW

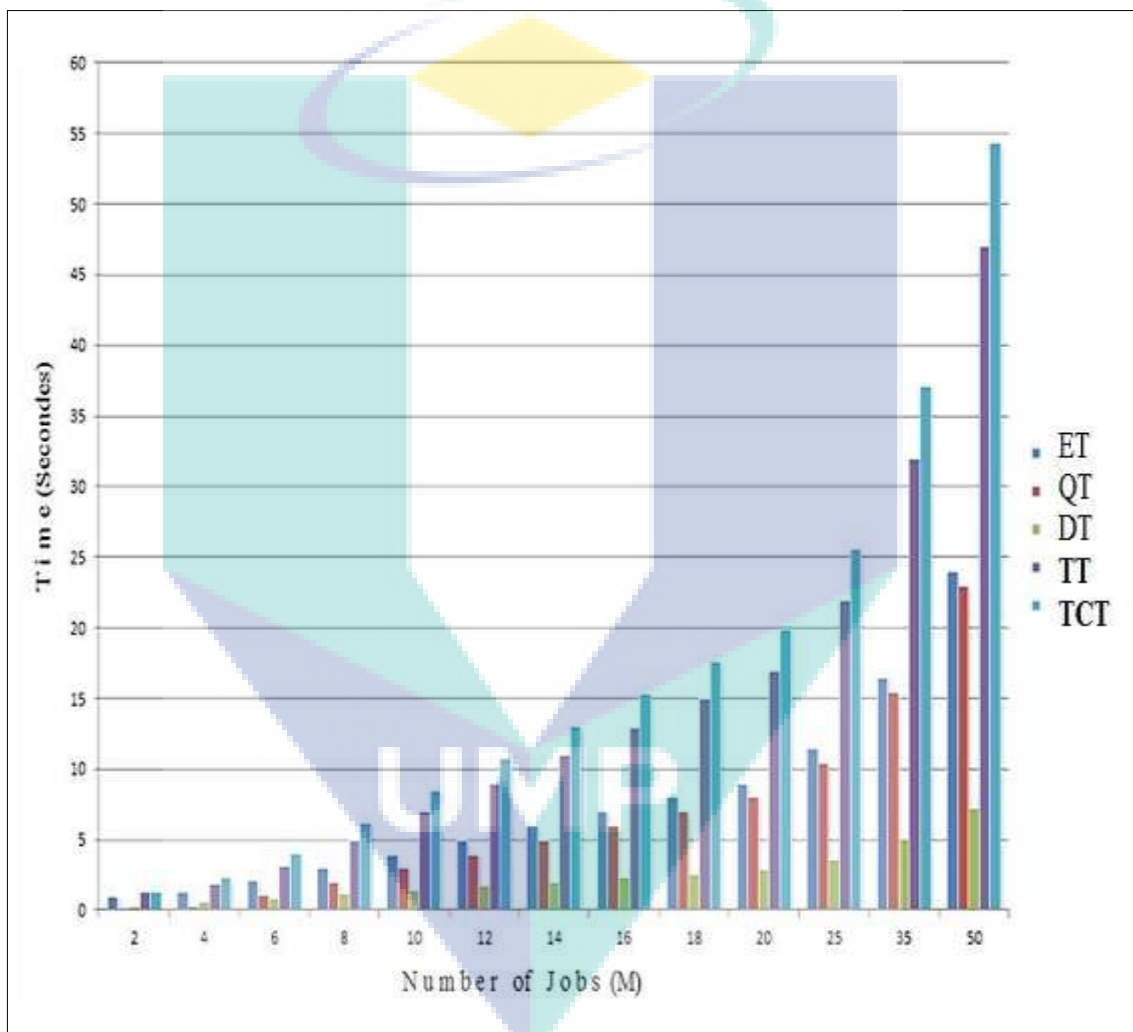
#### 4.8 Variation in TT and TCT with change in servers' number

This experiment has done for the comparison for times  $TT$  and  $TCT$ , to know the variation with changes in the number of servers ( $C$ ). Other parameters have the same value, only the number of servers has changed. The result shows the gap between both times  $TT$  and  $TCT$ . The gap between both *times* is less by using high bandwidth than by using low bandwidth. For  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 56Kbps$ , Figure 4.8 shows the comparison between  $TT$  and  $TCT$  with accuracy percentage.



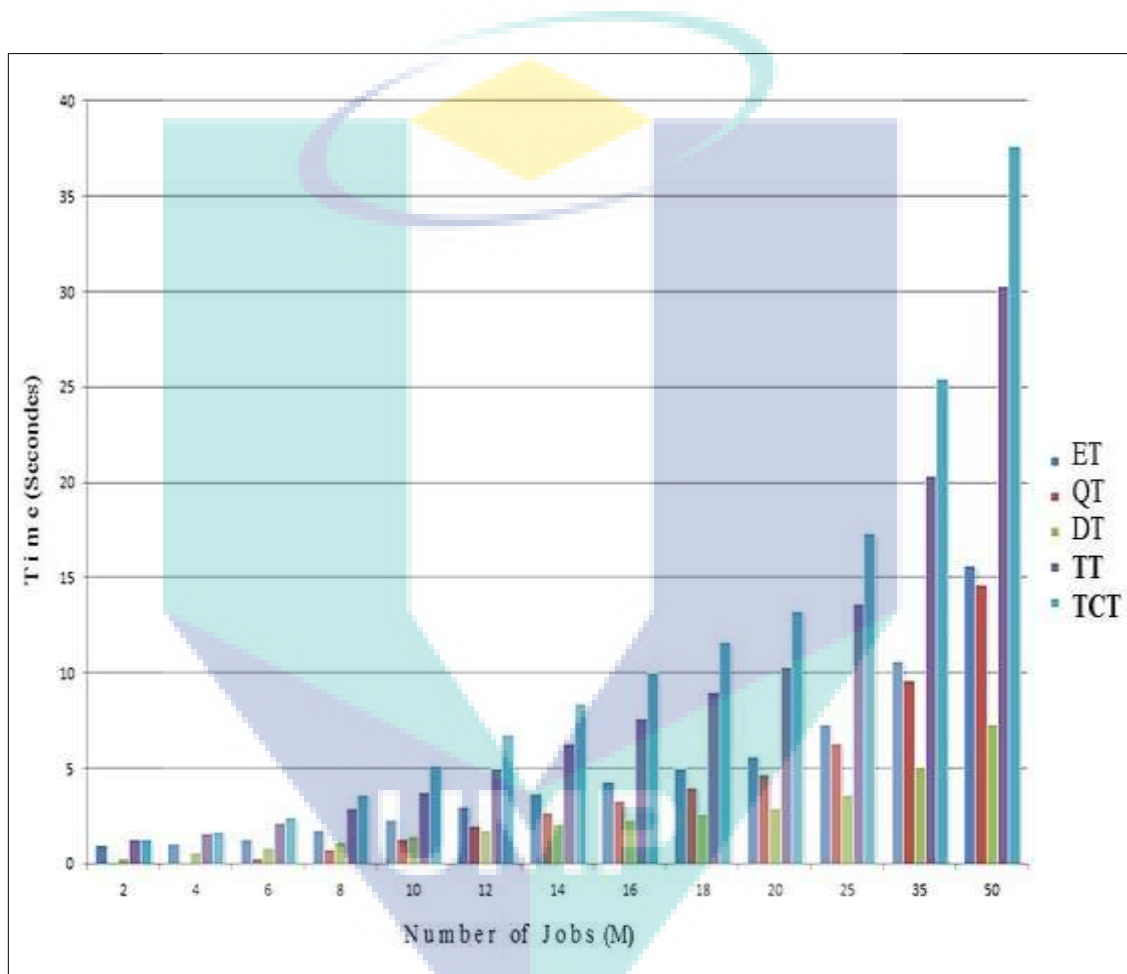
**Figure 4.8:** TT and TCT using single server

By using two servers and 56Kbps bandwidth, Figure 4.9 compares both times ( $TT$  and  $TCT$ ), and tries to know the variation due to the changes in the number of servers  $C$ . These results can be compared with Figure 4.9, three servers have been used. By using two servers,  $QT$  and  $ET$ , will decrease and as a result  $TT$  and  $TCT$  will automatically decrease. The mentioned gap between  $TT$  and  $TCT$  in Figure 4.9 is less than the gap in Figure 4.8. For  $C = 2$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 56Kbps$ .



**Figure 4.9:** TT and TCT using two servers

Figure 4.10 describes the difference between  $TT$  and  $TCT$  by using three servers and 56Kbps bandwidth with  $C = 3$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 56Kbps$ . It shows the effect of using three servers on  $TT$  and  $TCT$ .  $QT$  and  $ET$  will decrease and as a result,  $TT$  and  $TCT$  will automatically decrease. As mentioned above, there is no effect on  $DT$  by using more servers. The gap between  $TT$  and  $TCT$  in Figure 4.10 is decreasing as compared to the gap in Figure 4.9.

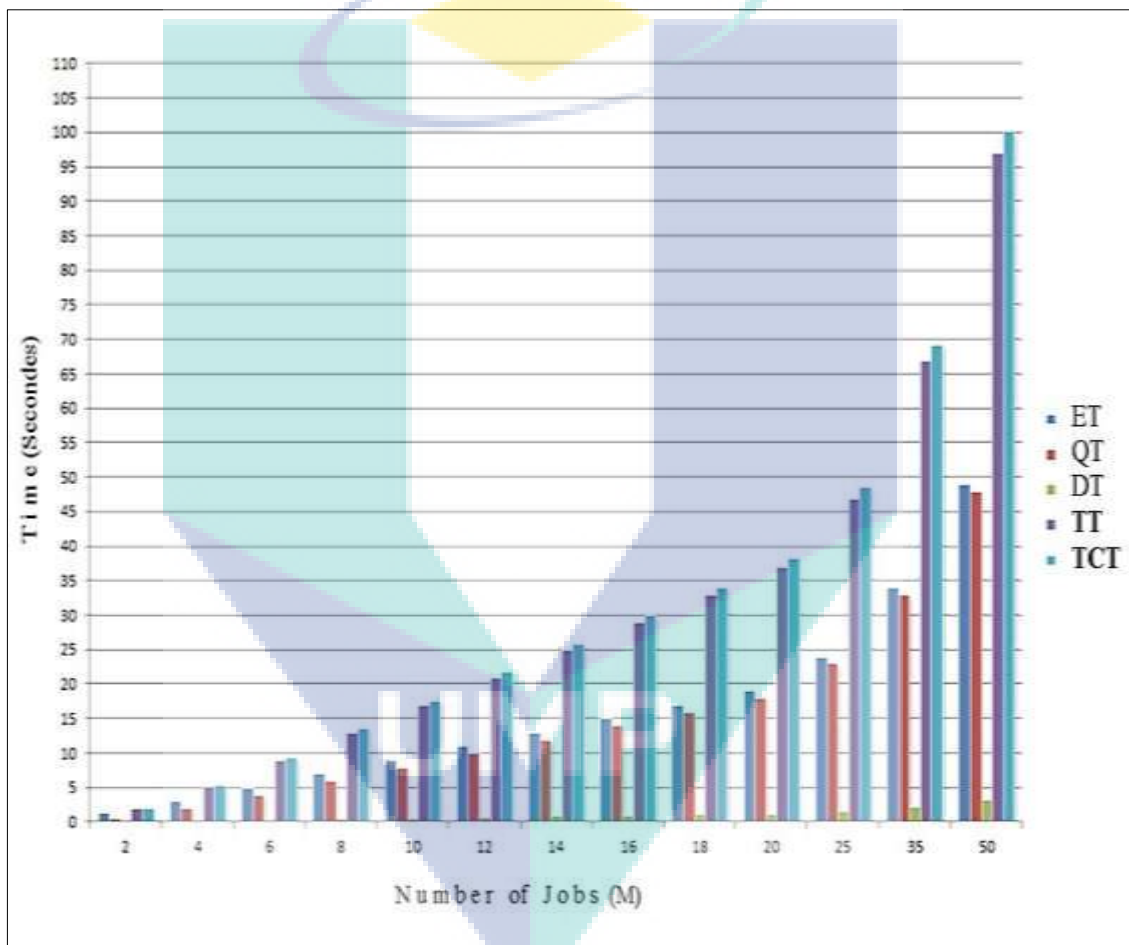


**Figure 4.10:** TT and TCT using three servers



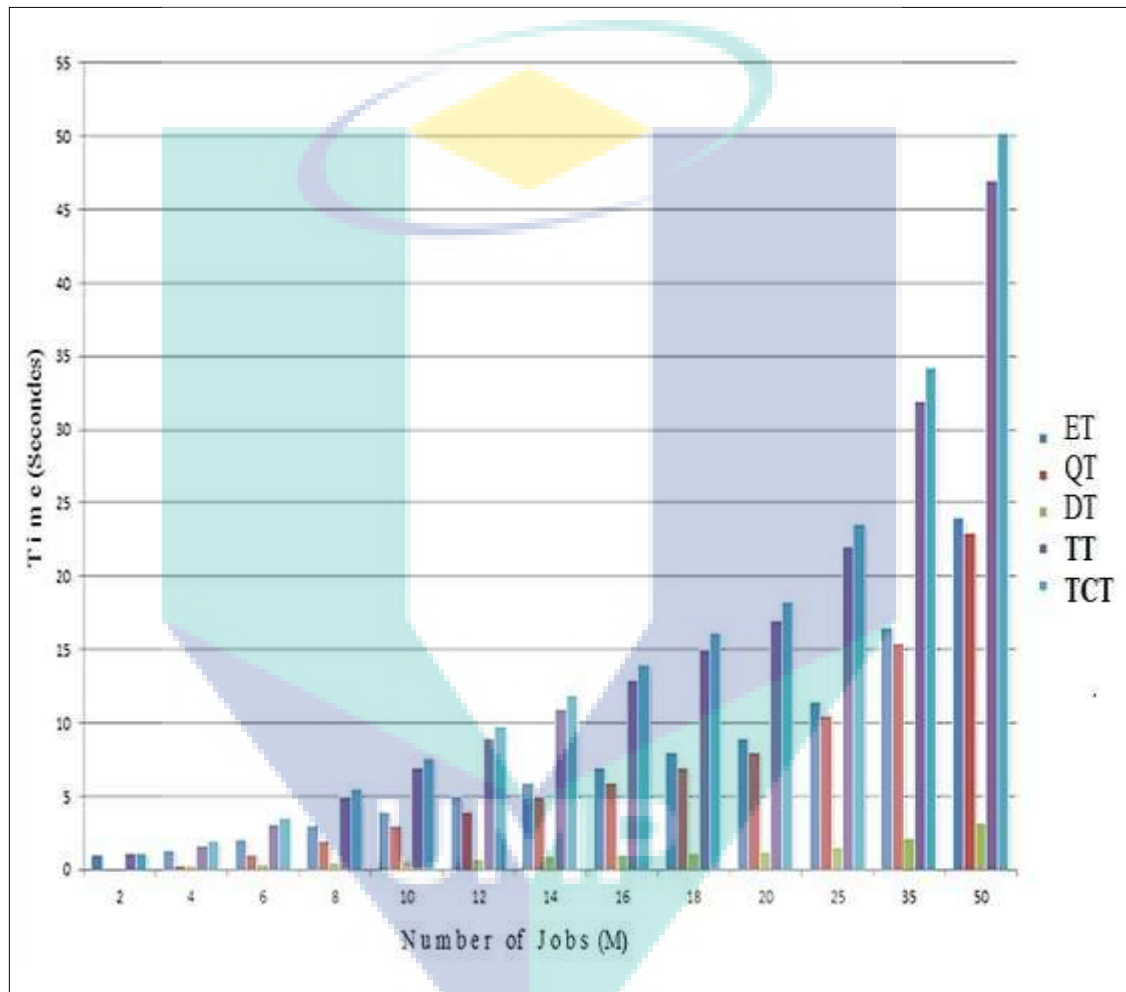
#### 4.9 TT and TCT variation by changing the servers' number with bandwidth

Figure 4.11 describes the difference between  $TT$  and  $TCT$  by using three servers and 128Kbps bandwidth. The result shows the effect with 3 servers.  $QT$  and  $ET$  will decrease and as a result,  $TT$  and  $TCT$  will automatically decrease. The gap between  $TT$  and  $TCT$  in Figure 4.11 has decreased, as compared to the gap in Figure 4.10; the used parameters are  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 128Kbps$ .



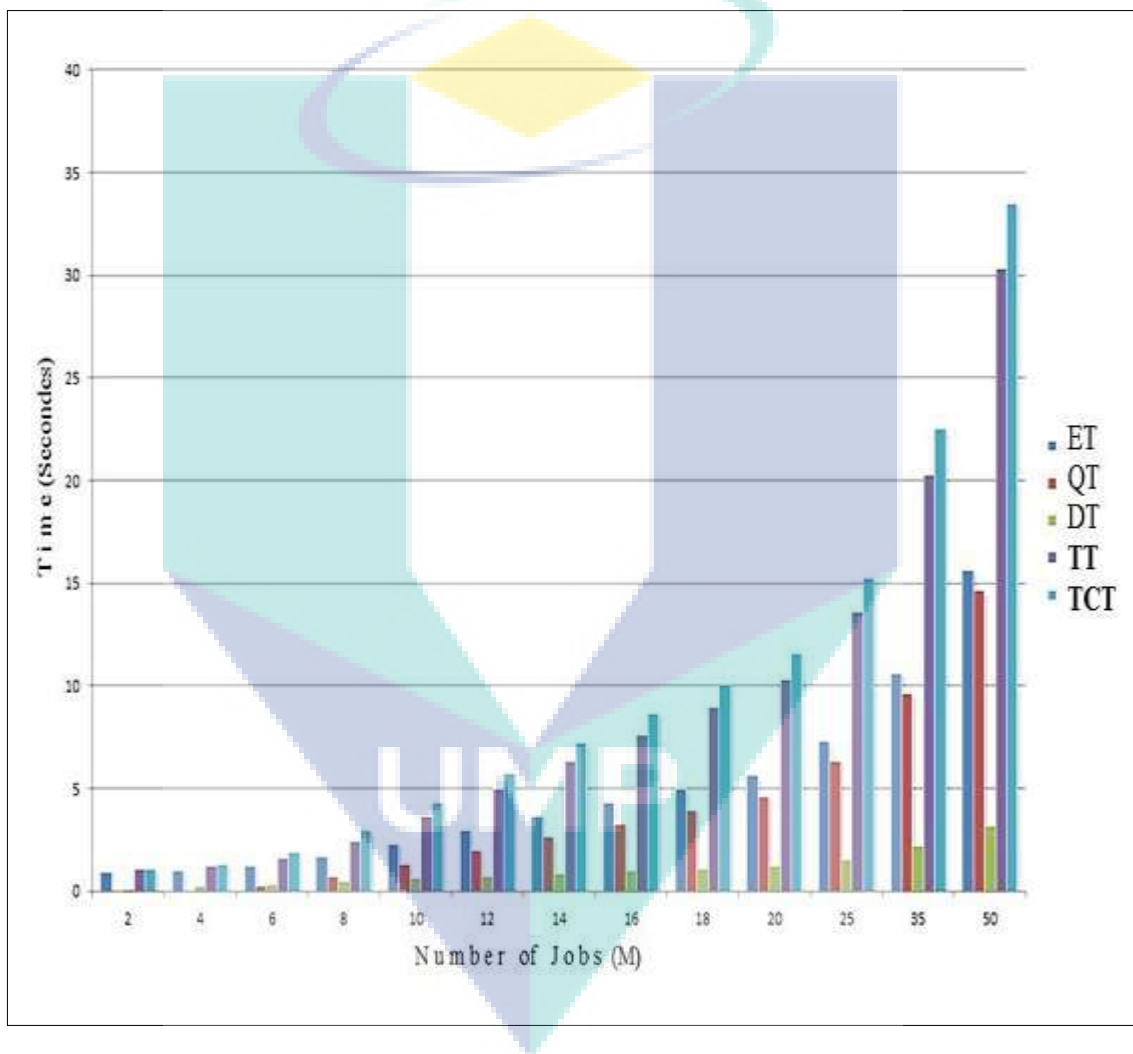
**Figure 4.11:** TT and TCT by a single server with 128Kbps  $BW$

By using  $C = 2$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 128Kbps$ , Figure 4.10 compares the difference between  $TT$  and  $TCT$  by using two servers and 128Kbps bandwidth.  $QT$  and  $ET$  are decreasing continuously, and as a result,  $TT$  and  $TCT$  are automatically decreasing. The gap between  $TT$  and  $TCT$  is also decreasing as compared to the gap shown in Figure 4.11, with a decreased ratio shown in Figure 4.12.



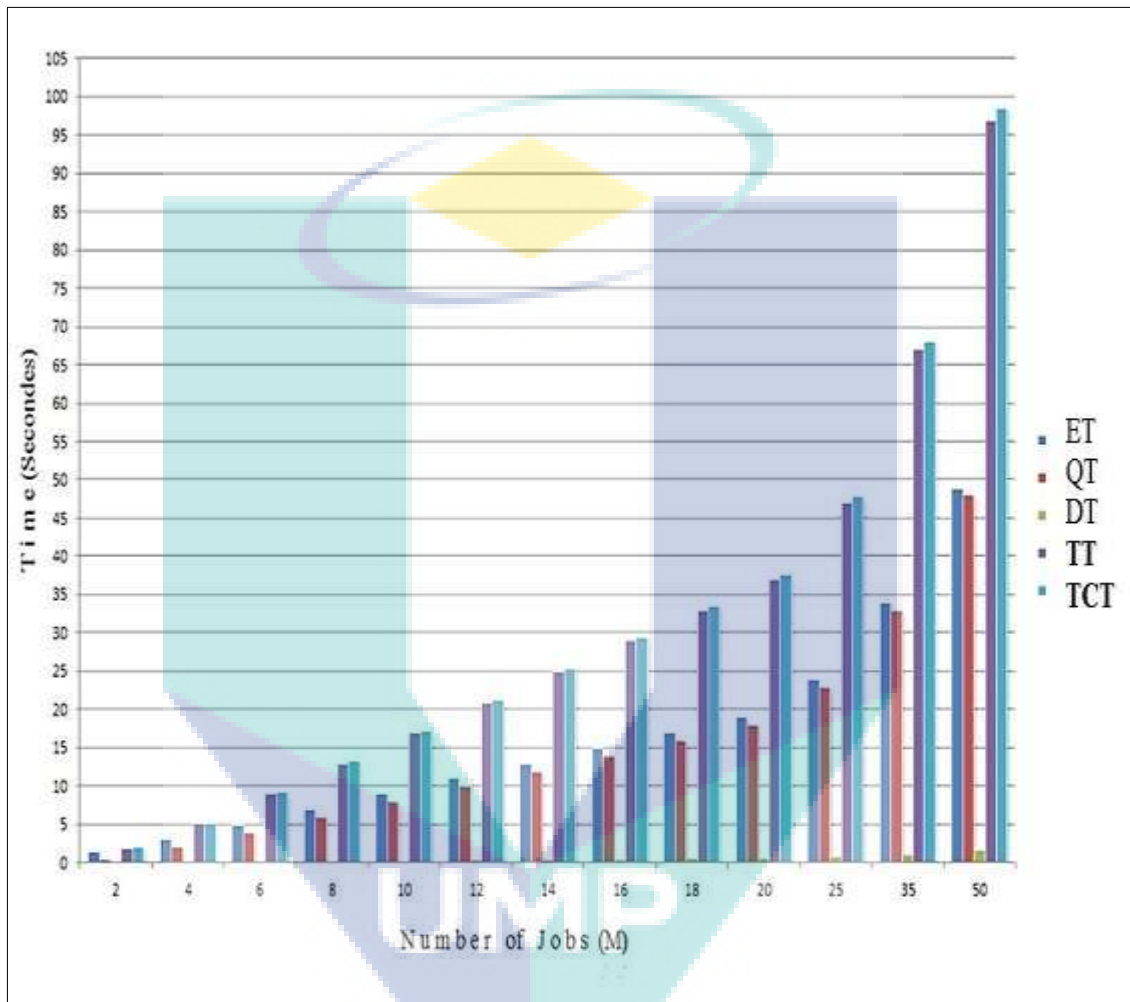
**Figure 4.12:** TT and TCT by two servers with 128Kbps  $BW$

Figure 4.11 shows the comparison between  $TT$  and  $TCT$  by using three servers with  $C = 3$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 128Kbps$ . Both times  $TT$  and  $TCT$  are decreasing with increase in the number of servers  $C$ . As mentioned above, there is no effect on  $DT$  by using three servers instead of one or two servers. The gap between  $TT$  and  $TCT$  in Figure 4.13 has decreased continuously as compared to the gap in Figure 4.11 and Figure 4.12.



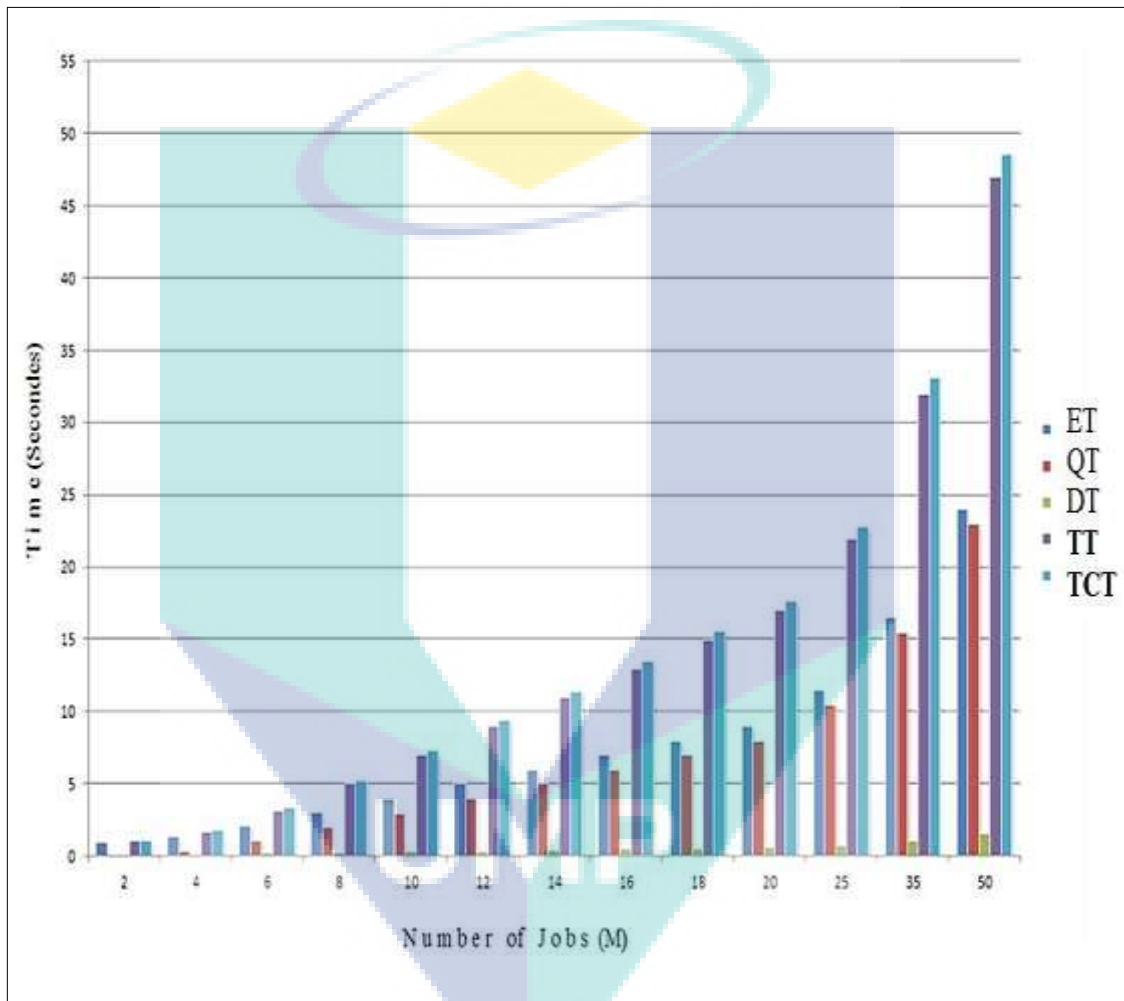
**Figure 4.13:** TT and TCT by three servers with 128Kbps  $BW$

By using  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 256Kbps$ , Figure 4.14 compares  $TT$  and  $TCT$  by using one server and 256Kbps bandwidth.  $TT$  and  $TCT$  are increasing while the difference between both *times* is decreasing with an increase in population and bandwidth.



**Figure 4.14:** TT and TCT by a single server with 256Kbps  $BW$

Figure 4.15 shows the difference between  $TT$  and  $TCT$  by using two servers and 256Kbps bandwidth. As mentioned above, there is no effect on  $DT$  by using two servers instead of one. For  $C = 2$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 256\text{Kbps}$ ,  $TT$  and  $TCT$  are increasing while the difference between both *times* is decreasing with an increase in population and bandwidth.



**Figure 4.15:** TT and TCT by two servers with 256Kbps  $BW$

By using  $C = 3$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 256Kbps$ , Figure 4.16 compares  $TT$  and  $TCT$  by using three servers and 256Kbps bandwidth. The gap between  $TT$  and  $TCT$  is decreasing with increase in number of jobs.

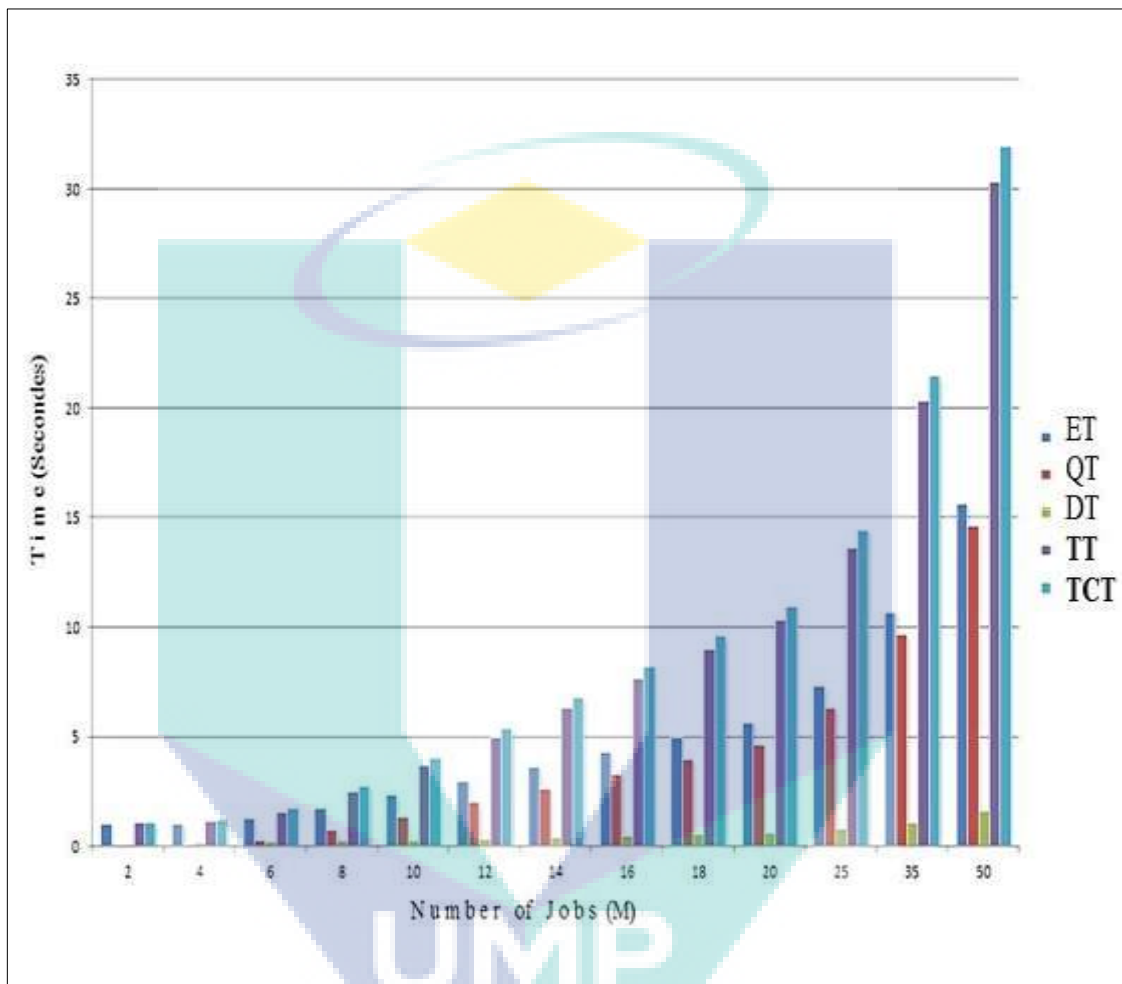


Figure 4.16: TT and TCT by three servers with 256Kbps BW

Figure 4.17 shows the comparison between  $TT$  and  $TCT$  by using a single server and 512Kbps bandwidth. There is a big change in  $DT$  by using 512Kbps because the bandwidth change can affect  $DT$  only. For  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 512Kbps$ , the gap between  $TT$  and  $TCT$  is decreasing with increase in population and bandwidth.

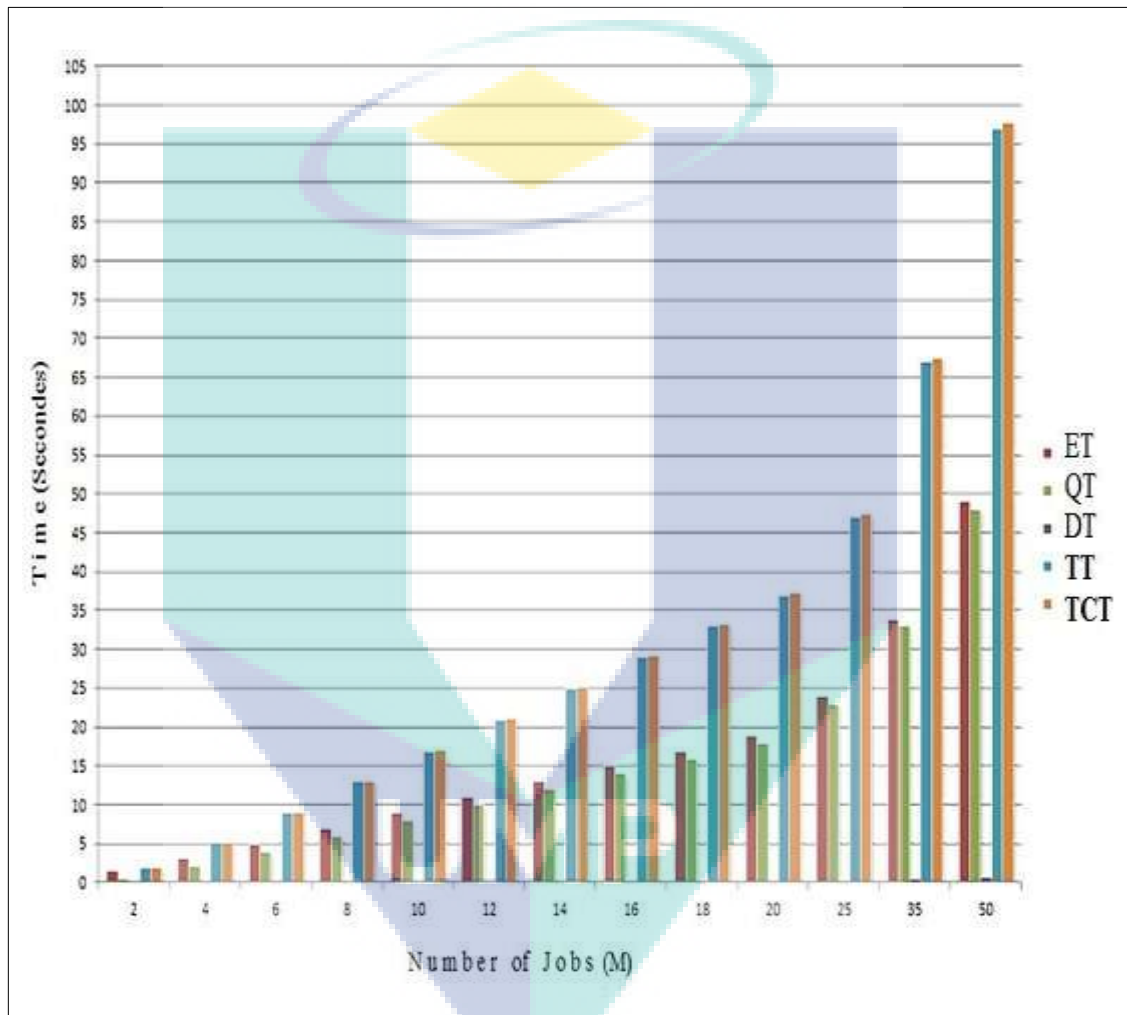
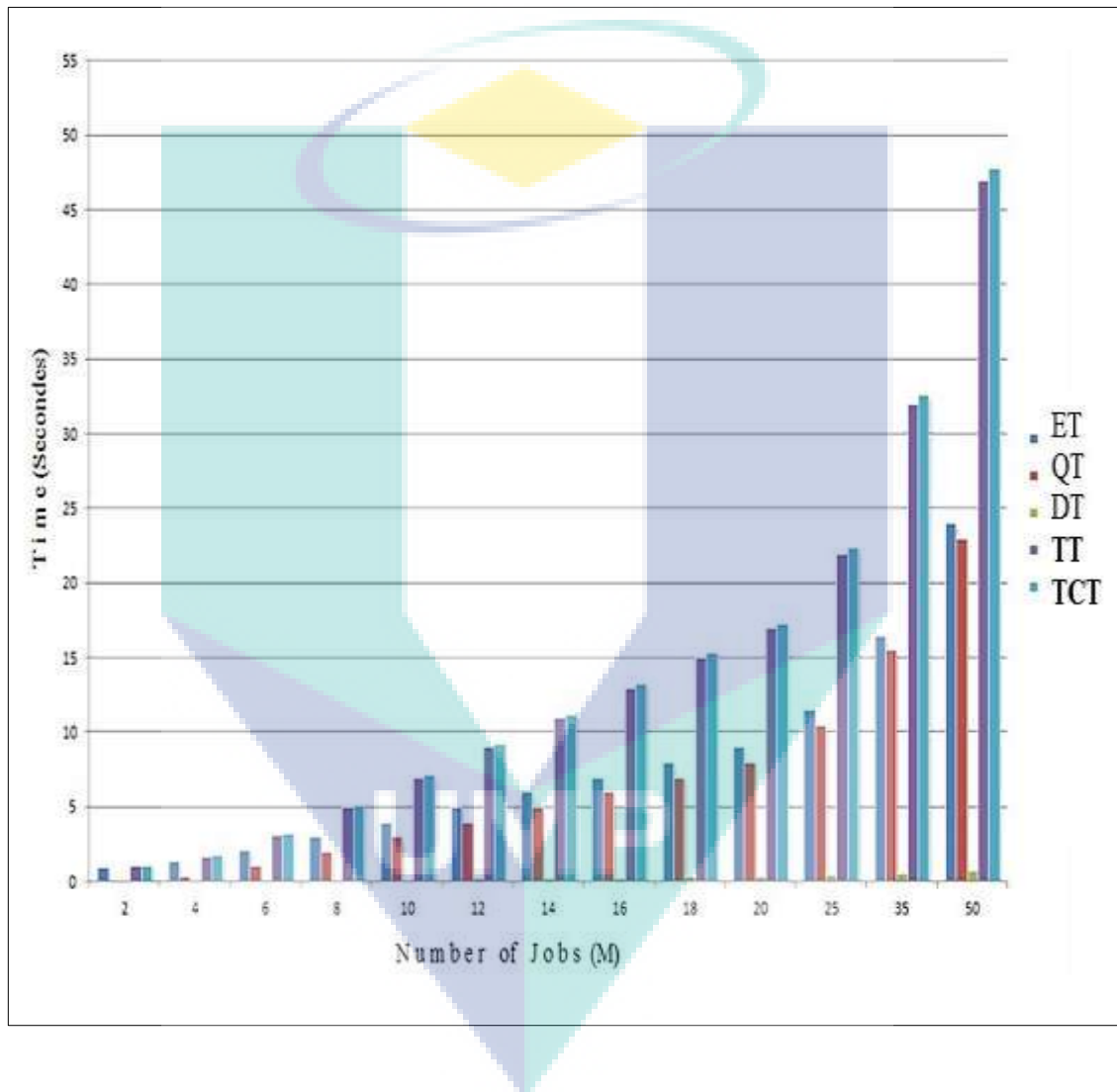


Figure 4.17: TT and TCT variation with  $C = 1$ ,  $BW = 512Kbps$

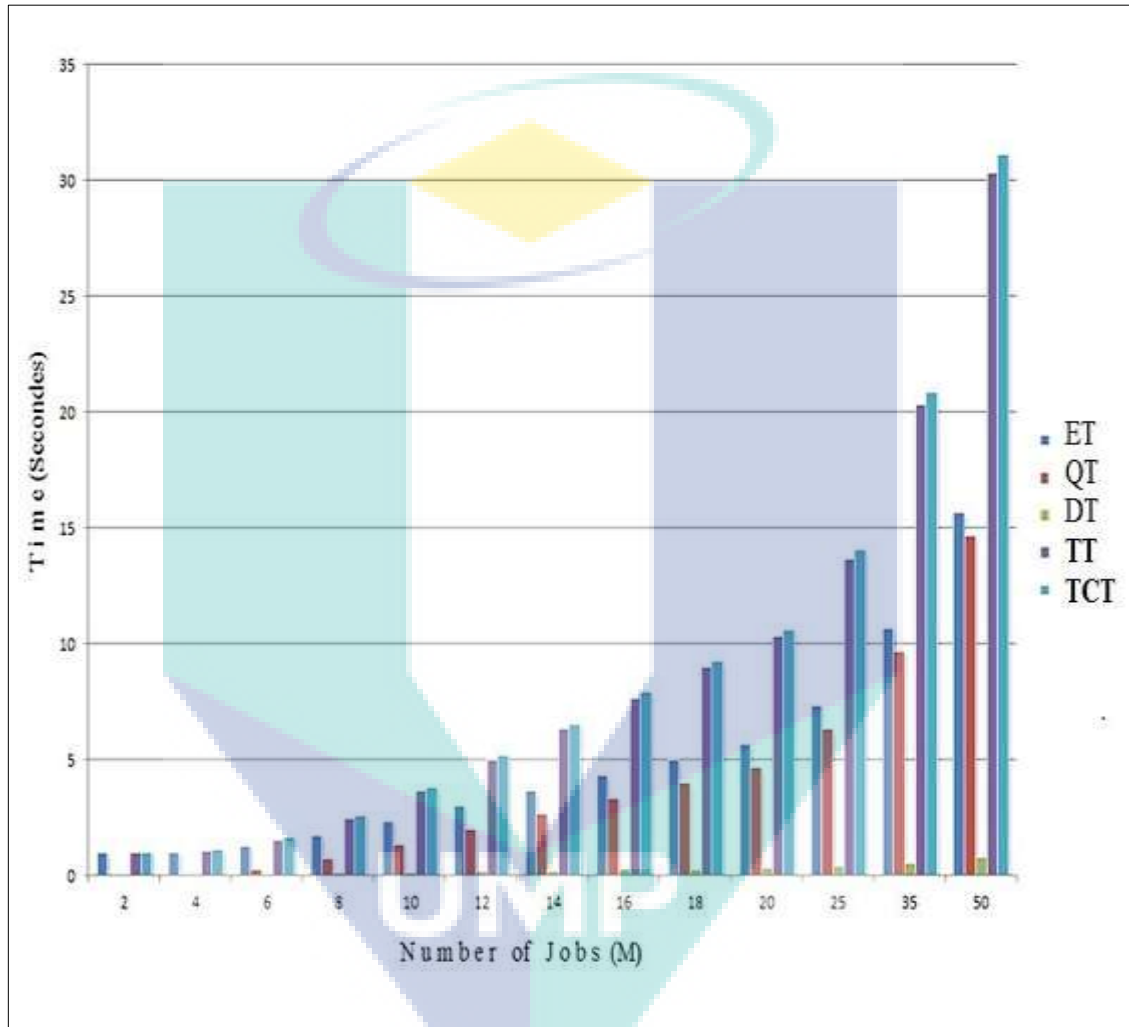
By using  $C = 2$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 512Kbps$ , Figure 4.18 shows the comparison between  $TT$  and  $TCT$  by using two servers instead of one. Increase in the number of servers directly affects the Execution Time  $ET$ , due to which the gap between both *times* is decreasing continuously with an increase in population and bandwidth.



**Figure 4.18:** TT and TCT variation with  $C = 2$ ,  $BW = 512Kbps$



Figure 4.19 compares  $TT$  and  $TCT$  by using three servers and 512Kbps bandwidth. There is no change in  $DT$ . For  $C = 3$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2 \sim 50$ ,  $BW = 512Kbps$ , as mentioned above, gap between  $TT$  and  $TCT$  is decreasing with the increase in population and bandwidth.



**Figure 4.19:** TT and TCT variation with  $C = 3$ ,  $BW = 512Kbps$

#### 4.10 Error and Accuracy Estimation between TT and TCT

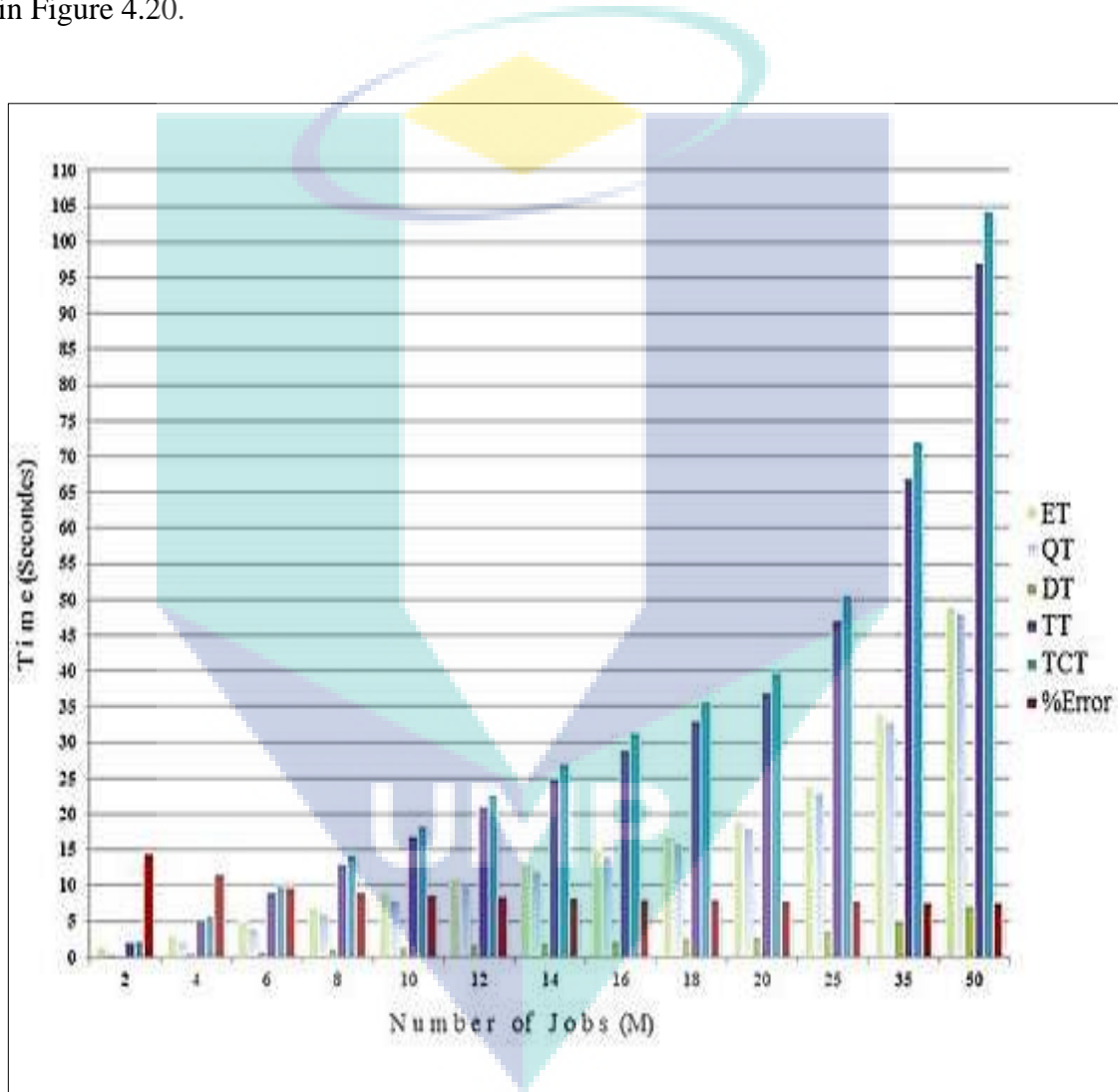
This section compares overall results, and calculates the variance between *Total Transfer Time (TT)* according to the existing technique and *Total Completion Time (TCT)* according to new proposed technique. Here actual value is *TT* and estimated value is *TCT*. The following general error estimation formula used by (Kani et al., 2010; Sankara, 2008; and Tang et al., 2006), to compare both times variations.

$$\% \text{ Error} = \{(\text{Estimate Value} - \text{Actual Value}) / \text{Actual}\} * 100$$

**Table 4.2:** Error Estimation with C = 1, BW = 56Kbps

Population	Wait in Queue	Wait in System	Transfer Time	Maximum of Wq & Tt	Total Completion Time Using Existing Formula	Total Completion Time Using New Propose Formula	Result Difference B/W Exist & New Formula	Error Estimation % error = (estimate - actual) / actual * 100
M	QT	ET	DT	Max(QT,DT)	TT	TCT	Diff	% Error
2	0.5000	1.5000	0.2900	0.5000	2.0000	2.2900	0.2900	14.5000
4	2.0625	3.0625	0.5900	2.0625	5.1250	5.7150	0.5900	11.5122
6	4.0031	5.0031	0.8800	4.0031	9.0062	9.8862	0.8800	9.7710
8	6.0001	7.0001	1.1700	6.0001	13.0002	14.1702	1.1700	8.9999
10	8.0000	9.0000	1.4600	8.0000	17.0000	18.4600	1.4600	8.5882
12	10.0000	11.0000	1.7600	10.0000	21.0000	22.7600	1.7600	8.3810
14	12.0000	13.0000	2.0500	12.0000	25.0000	27.0500	2.0500	8.2000
16	14.0000	15.0000	2.3400	14.0000	29.0000	31.3400	2.3400	8.0690
18	16.0000	17.0000	2.6300	16.0000	33.0000	35.6300	2.6300	7.9697
20	18.0000	19.0000	2.9300	18.0000	37.0000	39.9300	2.9300	7.9189
25	23.0000	24.0000	3.6600	23.0000	47.0000	50.6600	3.6600	7.7872
35	33.0000	34.0000	5.1200	33.0000	67.0000	72.1200	5.1200	7.6418
50	48.0000	49.0000	7.3100	48.0000	97.0000	104.3100	7.3100	7.5361

Using  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2$ ,  $BW = 56Kbps$ , the accuracy is 85%; it is increased up to 92.4639% by using number of job range  $M = 2 \sim 50$ . The result shows that with the increase of population ( $M$ ) by using the same bandwidth and same number of server, the gap has decreased between the results of both techniques ( $TT$  &  $TCT$ ). Decreasing the gap means the accuracy is increased by using the new techniques. By using  $M > 500$ , *stability point* (where accuracy is 100%) can be achieved. As described in Figure 4.20.



**Figure 4.20:** Error Estimation with  $C = 1$ ,  $BW = 512Kbps$

Using  $C = 1$ ,  $\lambda = 1$ ,  $\mu = 1$ ,  $M = 2$ ,  $BW = 512Kbps$ , the accuracy is 98.5000%; when  $M$  reaches to 50, accuracy is increasing up to 99.1753%, as shown in Table 4.23. The result shows that with the increase of population ( $M$ ) by using the same bandwidth and same number of server, the gap between the results of both techniques ( $TT$  &  $TCT$ ) is decreasing. Decreasing the gap means the accuracy is increased by using the new techniques. In this experiment also, by using  $M > 500$ , *stability point* (where accuracy is 100%) can be achieved. Hence new technique is more efficient when we need to transfer large amount of data.

**Table 4.3:** Accuracy Estimation with  $C = 1$ ,  $BW = 512Kbps$

Population	Wait in Queue	Wait in System	Transfer Time	Total Completion Time Using Existing Formula	Total Completion Time Using New Propose Formula	Error Estimation % error = $\frac{(\text{estimate} - \text{actual})}{\text{actual}} * 100$	% Accuracy = $100 - \% \text{ Error}$
M	QT	ET	DT	TT	TCT	% Error	% Accuracy
2	0.5000	1.5000	0.0300	2.0000	2.0300	1.5000	98.5000
4	2.0625	3.0625	0.0600	5.1250	5.1850	1.1707	98.8293
6	4.0031	5.0031	0.1000	9.0062	9.1062	1.1103	98.8897
8	6.0001	7.0001	0.1300	13.0002	13.1302	1.0000	99.0000
10	8.0000	9.0000	0.1600	17.0000	17.1600	0.9412	99.0588
12	10.0000	11.0000	0.1900	21.0000	21.1900	0.9048	99.0952
14	12.0000	13.0000	0.2200	25.0000	25.2200	0.8800	99.1200
16	14.0000	15.0000	0.2600	29.0000	29.2600	0.8966	99.1034
18	16.0000	17.0000	0.2900	33.0000	33.2900	0.8788	99.1212
20	18.0000	19.0000	0.3200	37.0000	37.3200	0.8649	99.1351
25	23.0000	24.0000	0.4000	47.0000	47.4000	0.8511	99.1489
35	33.0000	34.0000	0.5600	67.0000	67.5600	0.8358	99.1642
50	48.0000	49.0000	0.8000	97.0000	97.8000	0.8247	99.1753

#### 4.11 Overall overview

New proposed model is more efficient to transfer large data with high bandwidth, as shown in Table 4.4 for  $C = 1$ ,  $BW = 56 \sim 512$ ,  $M = 2 \sim 50$ .

**Table 4.4: Overall Overview**

BW	M	ET	QT	DT	TT	TCT	% Error	% Accuracy
56Kbps	2	1.5000	0.5000	0.2900	2.0000	2.2900	14.5000	85.5000
	4	3.0625	2.0625	0.5900	5.1250	5.7150	11.5122	88.4878
	6	5.0031	4.0031	0.8800	9.0062	9.8862	9.7710	90.2290
	8	7.0001	6.0001	1.1700	13.0002	14.1702	8.9999	91.0001
	10	9.0000	8.0000	1.4600	17.0000	18.4600	8.5882	91.4118
	12	11.0000	10.0000	1.7600	21.0000	22.7600	8.3810	91.6190
	14	13.0000	12.0000	2.0500	25.0000	27.0500	8.2000	91.8000
	16	15.0000	14.0000	2.3400	29.0000	31.3400	8.0690	91.9310
	18	17.0000	16.0000	2.6300	33.0000	35.6300	7.9697	92.0303
	20	19.0000	18.0000	2.9300	37.0000	39.9300	7.9189	92.0811
	25	24.0000	23.0000	3.6600	47.0000	50.6600	7.7872	92.2128
35	34.0000	33.0000	5.1200	67.0000	72.1200	7.6418	92.3582	
50	49.0000	48.0000	7.3100	97.0000	104.3100	7.5361	92.4639	
128Kbps	2	1.5000	0.5000	0.1300	2.0000	2.1300	6.5000	93.5000
	4	3.0625	2.0625	0.2600	5.1250	5.3850	5.0732	94.9268
	6	5.0031	4.0031	0.3800	9.0062	9.3862	4.2193	95.7807
	8	7.0001	6.0001	0.5100	13.0002	13.5102	3.9230	96.0770
	10	9.0000	8.0000	0.6400	17.0000	17.6400	3.7647	96.2353
	12	11.0000	10.0000	0.7700	21.0000	21.7700	3.6667	96.3333
	14	13.0000	12.0000	0.9000	25.0000	25.9000	3.6000	96.4000
	16	15.0000	14.0000	1.0200	29.0000	30.0200	3.5172	96.4828
	18	17.0000	16.0000	1.1500	33.0000	34.1500	3.4848	96.5152
	20	19.0000	18.0000	1.2800	37.0000	38.2800	3.4595	96.5405
	25	24.0000	23.0000	1.6000	47.0000	48.6000	3.4043	96.5957
35	34.0000	33.0000	2.2400	67.0000	69.2400	3.3433	96.6567	
50	49.0000	48.0000	3.2000	97.0000	100.2000	3.2990	96.7010	
256Kbps	2	1.5000	0.5000	0.0600	2.0000	2.0600	3.0000	97.0000
	4	3.0625	2.0625	0.1300	5.1250	5.2550	2.5366	97.4634
	6	5.0031	4.0031	0.1900	9.0062	9.1962	2.1097	97.8903
	8	7.0001	6.0001	0.2600	13.0002	13.2602	2.0000	98.0000
	10	9.0000	8.0000	0.3200	17.0000	17.3200	1.8824	98.1176
	12	11.0000	10.0000	0.3800	21.0000	21.3800	1.8095	98.1905
	14	13.0000	12.0000	0.4500	25.0000	25.4500	1.8000	98.2000
	16	15.0000	14.0000	0.5100	29.0000	29.5100	1.7586	98.2414
	18	17.0000	16.0000	0.5800	33.0000	33.5800	1.7576	98.2424
	20	19.0000	18.0000	0.6400	37.0000	37.6400	1.7297	98.2703
	25	24.0000	23.0000	0.8000	47.0000	47.8000	1.7021	98.2979
35	34.0000	33.0000	1.1200	67.0000	68.1200	1.6716	98.3284	
50	49.0000	48.0000	1.6000	97.0000	98.6000	1.6495	98.3505	
512Kbps	2	1.5000	0.5000	0.0300	2.0000	2.0300	1.5000	98.5000
	4	3.0625	2.0625	0.0600	5.1250	5.1850	1.1707	98.8293
	6	5.0031	4.0031	0.1000	9.0062	9.1062	1.1103	98.8897
	8	7.0001	6.0001	0.1300	13.0002	13.1302	1.0000	99.0000
	10	9.0000	8.0000	0.1600	17.0000	17.1600	0.9412	99.0588
	12	11.0000	10.0000	0.1900	21.0000	21.1900	0.9048	99.0952
	14	13.0000	12.0000	0.2200	25.0000	25.2200	0.8800	99.1200
	16	15.0000	14.0000	0.2600	29.0000	29.2600	0.8966	99.1034
	18	17.0000	16.0000	0.2900	33.0000	33.2900	0.8788	99.1212
	20	19.0000	18.0000	0.3200	37.0000	37.3200	0.8649	99.1351
	25	24.0000	23.0000	0.4000	47.0000	47.4000	0.8511	99.1489
35	34.0000	33.0000	0.5600	67.0000	67.5600	0.8358	99.1642	
50	49.0000	48.0000	0.8000	97.0000	97.8000	0.8247	99.1753	

New proposed model is more efficient to transfer large data with high bandwidth, as shown in Table 4.4 for  $C = 1$ ,  $BW = 56 \sim 512$ ,  $M = 2 \sim 50$ . With increase of data size, error is decreasing by using various bandwidth in range 56 ~ 512. When error is decreasing by increasing bandwidth, it shows that maximum accuracy is possible to achieve by using high bandwidth for large amount of data.

#### 4.12 SUMMARY

Cloud Computing presents various services remotely everywhere from anywhere in the world. Efficient Cloud service's provision is the basis on the efficient data transferring in Cloud environment. Further, efficient data transfer is possible in the presence of efficient and accurate scheduling technique for data transfer.

In this chapter, scheduling technique *Total Transfer Time (TCT)* has been introduced to support the calculation process for Data transferring. In new scheduling technique, this study offers importance to each parameter while calculating the Total Completion Time. For the evaluation of new model,  $M/M/C/*P$  queuing model has been used. There is a great impact on accuracy by taking each parameter separately in the formula for the *Total Completion Time (TCT)* for a job. After comparison of results, this study shows that *TCT* can be the sum of *Wait in the queue (QT)*, *Wait in system (ET)*, and *Data Transfer time (DT)*. The results of the tests show that the new proposed scheduling technique is more efficient for large data transfer.

Using  $M = 2$ ,  $BW = 56Kbps$ , the accuracy is 85%; it is increased up to 92.4639% by changing population  $M$  upto 50, as shown in Figure 4.19. And by using  $M = 2$ ,  $BW = 512Kbps$ , the accuracy is 98.5000%; when  $M$  reaches to 50, accuracy is increasing up to 99.1753%, as shown in Table 4.23. Result shows that with the increase of population ( $M$ ) by using the same bandwidth and same number of server, the gap between both techniques ( $TT$  &  $TCT$ ) is decreasing. Decreasing the gap means the accuracy is increased by using the new techniques. In this experiment also, by using  $M > 500$ , *stability point* (where accuracy is 100%) can be achieved. Hence new technique is more efficient when we need to transfer large amount of data.

## CHAPTER 5

### CONCLUSION AND FUTURE WORKS

#### 5.1 Introduction

This work has been addressed using  $M/M/C/*P$  queuing to produce data to support the time calculation for data transfer from source to target. This is operated by the Cloud service provider to its users. This chapter summarizes the important findings from this research. It also includes some directions for future work in each of the areas covered during this study.

#### 5.2 Scientific & Technological Contributions

Scheduling technique has introduced to support the calculation for the Total Time of Data Transferring process. Efficient data transfer is possible in the presence of efficient and accurate scheduling technique for data transfer. There is a great impact on accuracy by taking each parameter separately in formula for the Total Time of Completion for a job.

Scheduling technique *Total Transfer Time (TCT)* has been introduced to support the calculation process for Data transferring. After comparison of results, this study shows that *TCT* can be the sum of *Wait in the queue (QT)*, *Wait in system (ET)*, and *Data Transfer time (DT)*. The results of the tests show that the new proposed scheduling technique is more efficient for large data transfer.

Using  $M = 2$ ,  $BW = 56Kbps$ , the accuracy is 85%; it is increased up to 92.4639% by changing population  $M$  upto 50, as shown in Figure 4.19. And by using  $M = 2$ ,  $BW = 512Kbps$ , the accuracy is 98.5000%; when  $M$  reaches to 50, accuracy is increasing up to 99.1753%, as shown in Table 4.23. Decreasing the gap (as shown in Table 4.4) means the accuracy is increased by using the new techniques. Experiment shows that by using  $M > 500$ , *stability point* (where accuracy is 100%) can be achieved. Hence new technique is more efficient when we need to transfer large amount of data.

### 5.3 Limitations

For the evaluation of the new technique only queuing  $M/M/C/K/P$  model has used. For Queue Time ( $QT$ ) and Execution Time ( $ET$ ) 1 job is equal to 1 Kb. Due to using  $M/M/C/K/P$ , Queue Capacity ( $K$ ) is not including in any simulation in proposed technique.

### 5.4 Conclusion

Dealing with the large amount of data makes the requirement more critical for accuracy, efficiency, and reliability in data access. In this research, the techniques based on previous works done by other researchers have been discussed in Chapter 2. This study has introduced an alternate scheduling technique to support the calculation process for the *Total Transferring Time* for Data. In scheduling techniques; this research highlights the importance to each parameter while calculating the Total Completion Time.

We have evaluated our new technique by using  $M/M/C/*P$  queuing models. There is a great impact on accuracy by taking each parameter separately in Equation 3.6 to calculate the Total Completion Time ( $TCT$ ) for a job as discussed in Chapter 3. Hence this research concludes that the *Total Transferring Time* for data can be the sum of *Wait in the queue* ( $QT$ ), *Wait in system* ( $ET$ ), and *Data Transfer Time* ( $DT$ ).



Experiments have been conducted in order to prove this technique which can preserve data transfer time calculation. Chapter 4 compares the results of existing (Equation 3.5) and new proposed techniques (Equation 3.17), with variation of server's number and with change in bandwidth. For servers range  $C = 1\sim 3$  has used and bandwidth range  $BW = 56Kbps \sim 512Kbps$  has applied.

Using  $C = 1, \lambda = 1, \mu = 1, M = 2, BW = 56Kbps$ , the accuracy is 85%; it is increased up to 92.4639% by using  $M = 50$ . For  $C = 1, \lambda = 1, \mu = 1, M = 2, BW = 512Kbps$ , the accuracy is 98.5000%; when  $M$  reaches to 50, accuracy will increase up to 99.1753%, as shown in Table 4.23. The result shows that with the increase of population ( $M$ ) by using the same bandwidth and same number of server, the gap has decreased between the results of both techniques ( $TT$  &  $TCT$ ). Decreasing the gap means the accuracy is increased by using the new techniques. By using  $M > 50$ , *stability point* (where accuracy is 100%) can be achieved. Hence new technique is more efficient when we need to transfer large amount of data.

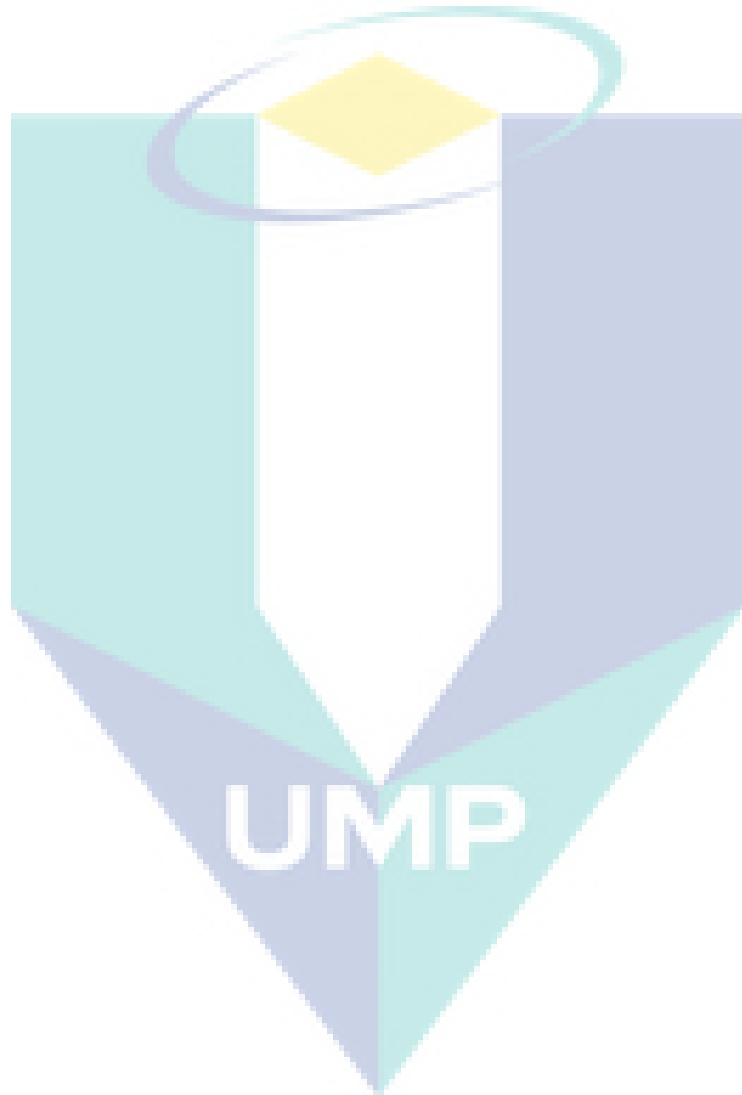
## 5.5 Future Work

Our future work will focus on data management and scheduling in Cloud Computing. Next plan is to investigate more with realistic scenarios by using other models. Additionally, to improve the efficiency in the provision of Cloud services, we would like to propose a complete real time model with the combination of replication, by using CloudSim. Further, we need to evaluate our new model by using other  $M/M/C$ ,  $M/M/Inf$ ,  $M/M/C/K$  and  $M/M/C/*M$  queuing models.

As the newer block for data storage, significant attention is receiving after outages in cloud services. Outages of service/s can directly impact the finances matters of cloud providers who are consistently looking for new ways to limit the reach and duration of outage events.

In order to keep active and alive Cloud Computing Services forever, it should be reachable for any one, reliable for each type of data, easy accessible from anywhere and

should be easy adoptable for any type of business. To achieve these goals, in future we plan to present architecture for locally sub-clouds instead of globally one cloud by using replication and scheduling with software agent. And to improve the efficiency in the provision of Cloud services, further study will propose a complete real time model with the combination of replication, by using CloudSim.



## REFERENCES

- Artalejo, J.R. and Lopez, M.J. 2007. A simulation study of a discrete-time multiserver retrial queue with finite population. *Journal of Statistical, Planning and Inference*. **137**(8): 2536-2542.
- Aggregate Knowledge. 2012. Making Media Accountable. <http://www.aggregateknowledge.com/index.html> (9 April 2012).
- Amazon. 2010. New amazon cloud outage takes down netflix. <http://www.crn.com/news/Cloud/231300459/new-amazon-Cloud-outage-takes-down-netflix-foursquare.htm?itc=refresh> (3 Jan 2011).
- Amazon EC2. 2010. Elastic Compute Cloud. <http://aws.amazon.com/ec2/> (25 April 2011).
- Armbrust, M., Armando, F., Rean, G., Anthony, D., Randy, H., Andrew, K., Gunho, L., David A., Stoica, I., and Zaharia, M. 2009. Above the clouds: A Berkeley view of cloud computing. *Technical Report No. UCB/EECS-2009-28*. <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf> (29 February 2012).
- Blanc, J.P.C. 2011. *Queuing Models: Analytical and numerical methods*. Netherland: Tilburg University.
- Buyya, R. 1999. *High performance cluster computing: Architectures and systems*. New York: Prentice Hall.
- Buyya, R., Broberg, J. and Goscinski, A. 2011. *Cloud Computing: Principals and Paradigms*. US and Canada: John Wiley & Sons.
- Buyya, R., Chee Shin Yeo, Srikumar Venugopala, James Broberg A. and Ivona Brandic. 2008. Cloud Computing and emerging it platforms: vision, hype, and reality for delivering Computing as the 5th utility. *Future generation computer system*. **25**(6): 599-616.
- Caron, E., Frederic, D., Adrian, M. 2010. Forecasting for grid and Cloud Computing on-demand resources based on pattern matching. *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom*. Indianapolis, Indiana, USA: 30 November – 3 December.

- Chee-Hock, N. and Soong B.H. 2008. *Queuing Modeling Fundamentals with Application in Communication Networks*. West Sussex, England: John Wiley & Sons.
- Cheng, Y. 2007. Two schedulers to provide delay proportion and reduce queuing delay simultaneously. *Journal of Computer Networks*. **51**(11): 3220–3231.
- Dattatreya, G.R. 2008. *Performance Analysis of Queuing and Computer Networks*. Sound Parkway NW: Tylor & Francis Group.
- Dean, J. and Ghemawat, S. 2004. Map Reduce: Simplified data processing on large clusters. *Proceeding of the Sixth Symposium on Operating System Design and Implementation (OSDI'04)*, pp. 137-149.
- Deelman, E. and Chervenak, A. 2008. Data Management Challenges of Data-Intensive Scientific Workflows. *Proceeding of the Eighth IEEE International Symposium on Cluster Computing and the Grid*, pp. 687-692.
- Delic, K.A. and Walker, M.A. 2008. Emergence of the academic computing clouds. *ACM Ubiquity Publication*, **9**(31): 1-4.
- Dwekat, Z. and Rouskas, G.N. 2011. A practical fair queuing scheduler: Simplification through quantization. *Journal of Computer Networks*. **55**(10): 2392-2406.
- Francini, A., Chiussi, F.M., Clancy, R.T., Drucker, K.D. and Idirence. N.E. 2001. Enhanced weighted round robin schedulers for accurate bandwidth distribution in packet networks. *Journal of Computer Network*. **37**(5): 561-578.
- Foster, I., Yong, Z., Loan, R. and Shiyong, L. 2008. Cloud Computing and Grid Computing 360-Degree Compared. *Proceeding of the IEEE Grid Computing Environments (GCE)*.
- Ghemawat, S., Gobiuff, H. and Leung, S.T. 2003. The Google file system. *Proceeding of the nineteenth ACM symposium on Operating systems principles (SOSP)*, pp. 29-43.
- Google App Engine Web Site. 2011. Web resource. <https://cloud.google.com/products/global> (Jan 7, 2011).
- Google. 2010. Google App Engine: Easy to build, Easy to scale, Easy to maintain. <http://code.google.com/appengine/> (03 May 2010).
- Gross, D. and Harris, C. M. 1985. *Fundamentals of Queuing Theory*. New York, US: John Wiley & Sons.
- Gross, D. 2008. *Fundamental of Queuing Theory*. New Jersey, USA: John Wiley & Sons.

- Gupta, U.C., Samanta, S.K., Sharma, R.K. 2004. Computing Queue Length and Waiting Time Distributions in Finite-Buffer Discrete-Time Multi-server Queue with Late and Early Arrivals. *International Journal of Computers & mathematics with Application*. **48**(10-11): 1557-1573.
- IBM. 2009. IBM Developer Works. <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0608mcinerney/index.html> (2 Jan 2011).
- Intro to Queuing Theory*. 2011. YouTube Video. <http://www.youtube.com/user/profbillbyrne>: Department of Management Studies, IIT Madras, India.
- Jain, J., Gopa, S., and Walter, B. 2007. *A Course on Queuing Models*. Sound Parkway NW, USA: Taylor & Francis Group.
- Jeremy, G., Elizabeth, W., Srinivas, R., Greg, O., and Bob, G. 2009. Twenty experts define cloud computing. *Cloud Expo Article*. [http://cloudcomputing.sys-con.com/read/612375\\_p.html](http://cloudcomputing.sys-con.com/read/612375_p.html) (24 August 2012).
- Jinhua, H., Jianhua, G., Guofei, S. and Tianhai, Z. 2010. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. *Proceeding of the Parallel Architectures, Algorithms and Programming (PAAP)*, pp. 89-96.
- John, D.C., Stephen, C. and Little. 2008. Little's Law. <http://web.mit.edu/sgraves/www/papers/Little%27s%20Law-Published.pdf> (16 Dec 2010).
- Kani, C., Shaojun, G., Yuanyuan, L., and Zhiliang, Y. 2010. Least Absolute Relative Error Estimation. *Journal of the American Statistical Association*. **105**(491): 1104-1112.
- Kaskade, J. 2009. Serial Entrepreneur Global Enterprise Executive. <http://jameskaskade.com> (27 March 2012).
- Kleinrock, L. 2005. A vision for the Internet. *International Journal of Research*. **2**(1): 4-5.
- Khan, N., Noraziah, A., Deris, M.M. and Ismail, E.I. 2011. Cloud Computing: Comparison of Various Features. *Proceeding of the Communications in Computer and Information Science*, pp. 243-254.
- Lec-30 Queuing Models*. 2010. YouTube Video. India: Department of Management Studies, IIT Madras.
- Lec-31 Single Server Queuing Models*. 2010. YouTube Video. YouTube Video. India: Department of Management Studies, IIT Madras.
- Lec-32 Multiple Server Queuing Models*. 2010. YouTube Video. YouTube Video. India: Department of Management Studies, IIT Madras.

- Liang, Z. and Shi, W. 2010. A reputation-driven scheduler for autonomic and sustainable resource sharing in Grid computing. *Journal of Parallel Distributed Computing*. **70**(2): 111–125.
- Lipsky, L. 2009. *Queuing Theory: A Linear Algebraic Approach*. New York, USA: Springer Science.
- Luis, R., Caron, E., Muresan, A., Desprez, F. 2012. Using clouds to scale grid resources: An economic model. *Future Generation Computer Systems*. **28**(4): 633-646.
- Mag, D.I., and Christian, D. 2009. *Stationary Queuing Models with Aspects of Customer Impatience and Retrial Behavior*. Vienna, Austria.
- Matias, W. 2008. Cloud Computing Taxonomy Map. <http://blogs.southworks.net/mwoloski/2008/08/19/cloud-computing-taxonomy-map> (9 March 2013).
- Nguyen, N. and Lim, S.B. 2007. Combination of Replication and Scheduling in Data Grids. *International Journal of Computer Science and Network Security*. **7**(3): pp. 304-308.
- OpenNebula. 2011. Enterprise cloud and datacenter virtualization. <http://www.opennebula.org> (10 January 2011).
- Peng, X. and Zhang, Z.L. 2009. Comparison of several Cloud Computing platforms. *Proceeding of The International Symposium Of Information*, pp. 23-27.
- Philippe, N. 1998. *Basic Elements of Queuing Theory: Application to the modeling of computer Systems*. Amherst, US: Informatical Society.
- Queuing Theory Calculator. 2009. Queuing Theory Model Calculator. <http://www.supositorio.com/rcalc/rcalclite.htm> (8 July 2011).
- Queuing Theory - Birth Death processes*. 2011. YouTube Video. YouTube Video. YouTube Video. India: Department of Management Studies, IIT Madras.
- Ranganathan, K. and Foster, I. 2003. Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids. *Journal of Grid Computing*. **1**(1): 53–62.
- Rimal, B.P., Eunmi, C. and Lumb, L. 2009. A Taxonomy and Survey of Cloud Computing System. *Proceeding of the Fifth International Joint Conference on INC, IMS and IDC*, pp. 44-51.
- Sankara, K. 2008. *Numerical Method for Scientists and Engineers*. New Delhi, India: Prentice-Hall, India.
- Say People. 2012. Cloud Computing. <http://saypeople.com/2011/08/15/cloud-computing/#axzz2HwQryoRi> (14 January, 2012).

- Search Cloud Application. 2013. Data as a service. <http://searchcloudapplications.techtarget.com/definition/data-as-a-service> (19 February 2012).
- Shi, Y., Xiaofeng, Z., and Xiangmei, L. 2010. Benchmarking cloud-based data management systems. *Proceedings of the Second International Workshop On Cloud Data Management*, pp. 47–54.
- Shailesh, S. 2011. *A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment*. California, US: San Jose State University Scholar Works.
- Stallings, W. 2000. *Queuing Analysis: A Practical Guide for Computer Scientists*. New York, US: IFORS Education Resources.
- T1 the complete telecom source. 2003. File Transfer Time. <http://www.t1shopper.com/tools/calculate/downloadcalculator.php> (9 July 2012).
- Tang, M., Lee, B.S. and Yeo, C.K. 2006. The Impact of data replication on job scheduling performance in the Data Grid. *Journal of Future Generation Computer Systems*. **22**(3): 254-268.
- Vaquero, L.M., Rodero-Merino, L., Caceres, J. and Lindner, M. 2009. A break in the Clouds: Towards a Cloud definition. *ACM SIGCOMM Computer Communication Review*. New York: ACM Press. **39**(1): 50-55.
- Vaishali, W., Sachin, D. and Nitin, A. 2012. An efficient data locality driven task scheduling algorithm for Cloud Computing. *International Journal of Multidisciplinary and Academic Research*. **1**(3): 1-8.
- Weiss, A. 2007. Computing in the Clouds. *ACM Networker*. New York, USA: NetWorker. **11**(4): 16-25.
- Zafril, R. and Azmi, M. 2011. Grid jobs scheduling improvement using priority rules and backfilling. *Proceeding of International Conference on Software Engineering and Computer Systems*, **179**: pp. 401– 415.

## AUTHOR'S BIODATA

**Nawsher Khan** was born in Samarbagh, Dir Lower, Pakistan. He received his bachelor degree in Computer Science from the University of Peshawar, Pakistan and Master Degree in Computer Science from Hazara University, Pakistan. Since 1999, he has worked in various educational institutions. In 2005, he was appointed in National Database and Registration Authority (NADRA) under the Interior Ministry of Pakistan. Currently he is a Research Assistant in the Faculty of Computer Systems and Software Engineering at University Malaysia Pahang (UMP), and before completion of PhD study, he got offer from University Malaya (UM) Kuala Lumpur as a Post Doctorate Research Fellow.

His research interests include Cloud Computing, Data Management, Distributed System, Scheduling, and Replication. He has published more than 25 articles in various international journals and conference proceedings.

The logo of University Malaysia Pahang (UMP) is a large, downward-pointing triangle composed of four smaller triangles meeting at the center. The top-left triangle is light blue, the top-right is light purple, the bottom-left is light green, and the bottom-right is light blue. The letters 'UMP' are written in white, bold, sans-serif font across the center of the triangle.

UMP



# PUBLICATION LIST

## INTERNATIONAL JOURNALS

1. **Nawsher Khan**, A. Noraziah, and Tutut Herawan. A Scheduling Technique for Data Transfer Time Calculation in Cloud and Grid Environment. To appear in International Journal of High Performance Computing Applications. ISI Impact Factor: 0.643. (Special Issue of ICICA 2012, Accepted).
2. **Nawsher Khan**, A. Noraziah, and Tutut Herawan. A Scheduling Architecture for Data Transfer Time Calculation. Journal of Parallel and Distributed Computing, Elsevier, 2012. ISI Impact Factor 0.859 (Under Review)
3. **Nawsher Khan**, A. Noraziah, and Tutut Herawan. CLOUD COMPUTING: Locally Sub-Clouds instead of Globally one Cloud. IGI Global. International Journal of Cloud Applications and Computing, 2(3), pp. 68-84, July-September 2012.
4. **Nawsher Khan**, A. Noraziah, Elrasheed I. Ismail, and Mustafa Mat Deris. CLOUD COMPUTING: Analysis of Various Platforms. International Journal of E-Entrepreneurship and Innovation, IGI Global, 3(2), pp. 52-60, April-June 2012.
5. **Nawsher Khan**, Noraziah Ahmad, Ahmed N.A. Alla, and Abul H. Beg. A Novel Database Design for Student Information System. American Journal of Computer Science, Science Publication, Volume 6, Issue 1, 43-46, January 31, 2010.

## BOOK CHAPTERS

6. **Nawsher Khan**, A. Noraziah, Tutut Herawan, and Mustafa Mat Deris. Cloud Computing: Analysis of Various Services. ICICA 2012, Lecture Notes in Computer Science, Springer Verlag, Vol. 7473, Pages 397-404, 2012. (SCIMago/Scopus Impact Factor = 0.212)

7. **Nawsher Khan**, A. Noraziah, and Tutut Herawan. A Cloud Architecture with Efficient Scheduling Technique. ICICA 2012: Lecture Notes in Computer Science, Springer Verlag, Vol. 7473, Pages 381-388, 2012. (SCIMago/ScopusImpact Factor = 0.212)
8. **Nawsher Khan**, A. Noraziah, Mustafa Mat Deris, and Elrasheed I.Ismail. Cloud Computing: Comparison of Various Features. In E. Ariwa and E. Qawasmeh (Eds.): DEIS 2011, Communications in Computer and Information Science, Springer Verlag, Vol. 194, pp. 243–254, 2011.

## INTERNATIONAL CONFERENCES

9. **Nawsher Khan**, A. Noraziah, Tutut Herawan, Elrasheed Ismail. Cloud Computing: Architecture for efficient provision of services. The 15th International Conference on Network-Based Information Systems (NBIS-2012), Melbourne, Australia, 2012. Pages 18-23.
10. **Nawsher Khan**, A. Noraziah, Elrasheed I.Ismail, Mustafa Mat Deris. Cloud Computing: Analysis of Open-Source Platforms Features. To appear in Proceedia Computer Science Journal, Elsevier, Vol. x, pp. xxx-xxx, 2012.
11. **Nawsher Khan**; bt Ahmad, N.; Beg, A.H.; Fakheraldin, M.A.I.; Alla, A.N.A.; Nubli, M.; 2010. Mental and Spiritual Relaxation by Recitation of the Holy Quran. Computer Research and Development, 2010 Second International Conference on Digital Object Identifier: 10.1109/ICCRD. 2010.62 Publication Year: 2010, Page(s): 863 – 867.
12. **Nawsher Khan**.; Ahmad, N.; Beg, A.H.; Ismail, E.I.; Abd Alla, A.N.; Nubli, M.; 2010. Epilepsy Control by Prayer Type Yoga Exercise. *Computer Research and Development*, 2010 Second International Conference on Digital Object Identifier: 10.1109/ICCRD.2010.61 Publication Year: 2010, Page(s): 391 – 395.
13. **Nawsher Khan**, A.Noraziah, Elrasheed I.Ismail, Mustafa Mat Deris.*Cloud Computing: Analysis of Various Platforms Features. 2<sup>nd</sup> World Conference on Information Technology (WCIT 11)*, 24-27 November, 2011. Antalya, Turkey.

## AS A CO-AUTHOR PUBLICATION

14. Noraziah, Ainul Azila Che Fauzi, Mustafa Mat Deris, Md Yazid Mohd Saman, Noriyani Mohd Zain, **Nawsher Khan**, “Managing Educational Resource Student Information Systems Using BVAGQ Fragmented Database Replication Model”, Elsevier, *Procedia Social and Behavioural Sciences* 2013.
15. Elrasheed, I.S; Hatim, R; Noraziah Ahmad; **Nawsher Khan.**; Beg, A.H.; 2010. Effective Knowledge Management System Architecture in Cloud Computing. *2nd International Conference on Applied Business and Economics, ICABE 2010*, Sohar University Oman.
16. Elrasheed. I. Sultan,A. Noraziah, **Nawsher Khan**. Scheduling Based Load Balancing Using QPSO in Cloud Computing. *International Journal of Cloud Computing Application and Computing*, IGI Global, 2012. (In progress).
17. Elrasheed. I.Sultan,A. Noraziah, **Nawsher Khan**, Ainul AzilaCheFauzi, Noriyani M Zain, Gamal Awad. Task Scheduling Based on Quantum Particles Swarm Optimization Load Balancing in Cloud Computing, 2012. (Accepted)
18. Ainul Azila Che Fauzi, A. Noraziah, Noriyani M Zain, A.H. Beg, **Nawsher Khan** and Elrasheed Ismail Sultan. Handling Fragmented Database Replication through Binary Vote Assignment Grid Quorum. *Journal of Computer Science* DOI: 10.3844/jcssp.2011.1338.1342, Volume 7, Issue 9 Pages 1338-1342.
19. A.H.Beg, Noraziah Ahmad, Ahmed N Abd Alla, **Nawsher Khan**, Framework of Persistence Layer for Synchronous Data Replication (PSR). *Australian Journal of Basic and Applied Sciences*. 4(10): 5394-5400, 2010, ISSN 1991-8178, INSInet Publication.
20. A.H.Beg, Noraziah Ahmad, Ahmed N Abd Alla, **Nawsher Khan**, K.F. Rabbi. Structure and Framework of Synchronous Replication Based On Data Persistency to Improve Data Availability into a Heterogeneous System, 2010. *International Conference on Software and Computing Technology (ICSC) 2010*, Kunming, China.

21. Muhammad Khan, Salah A.A. Elhoussein, Muhammad Mumtaz Khan, **Nawsher Khan**. Anti-Acetyl cholinesterase Activity of Piper sarmentosum by a Continuous Immobilized-enzyme Assay. *3rd International Conference on Biotechnology and Food Science (ICBFS 2012)*, April 7-8, 2012 APCBEE Procedia, Volume 2, 2012, Pages 199–204.
22. Beg, A.H.; Ahmad, N.; **Nawsher Khan.**; Alla, A.N.A.; Nubli, M.; Lovely, A.K.; 2010. Artificial intelligent strategy to control heart rate variability. *Electronic Computer Technology (ICECT), 2010 International Conference on Digital Object Identifier: 10.1109/ICECTECH.2010. 5479951 Publication Year: 2010, Page(s): 228 – 231.*
23. Noaziah Ahmad, A.H.Beg, Ahmad N Abd Alla, Muhammad Nubli, **Nawsher Khan**, Ainul Azila. Framework of intelligent game and software for improvement of learning performance based on heart rate variability. *Computer Engineering and Technology (ICCET). 2nd International Conference on 16-18 April 2010, Chengdu, China, IEE Explore, V4, pp. 383-386.*
24. Habib Shah, Rozaida Ghazali, Nazri Mohd Nawi, and **Nawsher Khan**. Boolean Function classification using Hybrid Ant Bee Colony Algorithm. *Journal on Computer Science & Computational Mathematics, JCSCM, Volume 2, Issue 11, November 2012. pp. 61-70.*

A large, semi-transparent watermark logo is centered on the page. It features a stylized 'U' and 'M' in light blue and purple, with the letters 'UIMP' in white below them, all enclosed within a dashed white border.

UIMP