

إقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان :

Efficient Load Balancing Algorithm in Cloud Computing

أقر أن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه
حيثما ورد، وإن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو
بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

Declaration

The work provided in this thesis, unless otherwise referenced, is the
researcher's own work, and not has been submitted elsewhere for any
other degree or qualification.

Student's : Hafiz Jabr Younis

Signature:

Date:4/4/2015

اسم الطالب: حافظ جبر يونس

التوقيع:



التاريخ:4/4/2015

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Islamic University – Gaza
Deanery of Post Graduate Studies
Faculty of Information Technology



الجامعة الإسلامية – غزة
عمادة الدراسات العليا
كلية تكنولوجيا المعلومات

Efficient Load Balancing Algorithm in Cloud Computing

Prepared by:

Hafiz Jabr Younis

Supervised By:

Dr. Alaa El Halees

**A Thesis Submitted as Partial Fulfillment of the Requirements for
the Degree of Master in Information Technology**

Feb., 2015



نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ حافظ جبر حافظ يونس لنيل درجة الماجستير في كلية تكنولوجيا المعلومات / قسم تكنولوجيا المعلومات وموضوعها:

إنشاء خوارزمية فعالة لتوزيع الأحمال للحوسبة السحابية

Efficient Load Balancing Algorithm in Cloud Computing

وبعد المناقشة التي تمت اليوم الثلاثاء 04 جمادى الآخر 1436 هـ، الموافق 2015/03/24م الساعة الحادية عشرة صباحاً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

.....	مشرفاً ورئيساً	أ.د. علاء مصطفى الهليس
.....	مناقشاً داخلياً	د. ربحي سليمان بركة
.....	مناقشاً خارجياً	د. محمد عبد اللطيف راضي

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية تكنولوجيا المعلومات / قسم تكنولوجيا المعلومات.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله و لزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق ،،،

مساعد نائب الرئيس للبحث العلمي و للدراسات العليا

أ.د. فؤاد علي العاجز



*To The Soul
of My
Father...*



Dr. Jabr Younes

Acknowledgement

*First of all I thank **Allah** for guiding me and taking care of me all the time.*

*I would like to express my profound sense of gratitude towards my guide **Dr. Alaa El hales**, Professor, Department of Information technology, for his able guidance, support and encouragement throughout the period this work was carried out. His readiness for consultation at all times, his educative comments, his concern and assistance even with practical things have been invaluable.*

*I would also like to convey my sincerest gratitude and indebtedness to the entire faculty members and staff of the Department of Information technology and for **Dr. Mohamed Radi** who bestowed their efforts and guidance at appropriate times without which it would have been very difficult on my part to finish the project work,*

*Very special thanks to my **Father** who spent all his life to make me happy and always the best. Thanks for your pray, for your patience, for helping, guiding and supporting me through all my life.*

*I wish to express my considerable gratitude to my dear **Mother**, my lovely **Wife** and my sweet **Children** for their patience, motivation and continue support.*

Abstract

Recently, cloud computing become a new global trend of computing. It is a modern style of using the power of Internet and wide area network (WAN) to offer resources remotely. It's a new solution and strategy to achieve high availability, flexibility, cost reduced and on demand scalability. However cloud computing has many challenges such as poor resource utilization which has deep impact in the performance of cloud computing. These problems arisen due to the huge amounts of information. So the need for efficient and powerful cloud computing load balancing algorithms is one of the most important issues in this area to improve the performance of cloud computing.

Many researchers proposed various load balancing and job scheduling algorithms in cloud computing, but there is still some inefficiency in the system performance and load still imbalance. Therefore, in this research we propose a load balancing algorithm to improve the performance and efficiency in heterogeneous cloud computing environment. We propose a hybrid algorithm based on randomization and greedy algorithm, it takes advantages of both random and greedy algorithms. The algorithm considers the current resource information and the CPU capacity factor to achieve the objectives. The hybrid algorithm has been evaluated and compared with other algorithms using CloudAnalyst simulator. The results showed improvements on average response time and on processing time by considering the current resource information and the CPU capacity factor compared with other algorithms, and this means the performance has improved.

Keywords: *Load Balancing, Cloud Computing, Virtual Machine, Virtualization, Cloud Analyst, Scheduling.*

Table of Contents

Acknowledgement	ii
Abstract.....	iii
Table of Contents.....	iv
List of Figures	vi
List of Tables	vii
List of Abbreviations	viii
Chapter 1 Introduction	2
1.1 Statement of the problem	3
1.2 Objectives	3
1.2.1 Main objective	3
1.2.2 Specific objectives.....	3
1.3 Research Methodology	3
1.4 Significance of the thesis	4
1.5 Scope and limitations.....	5
1.6 Research Organization	5
Chapter 2 Technical Background and Related Works.....	7
2.1 Technical Background	7
2.1.1 Cloud Computing	7
2.1.2 Cloud computing features	8
2.1.3 Cloud Service Model	9
2.1.4 Cloud Deployment Model.....	10
2.1.5 Virtualization.....	11
2.1.6 Cloud Computing Issues and Challenges	11
2.1.7 Cloudsim	21
2.2 Related works	27
2.3 Summary	31
Chapter 3 Proposed Algorithm	33
3.1 Proposed Algorithm	33
3.1.1 Description	35
3.1.2 Implementation	35

3.1.3	Pseudo code.....	36
3.1.4	Evaluation	38
3.2	Summary	38
Chapter 4	Experiments and Results.....	40
4.1	Experiments	40
4.1.1	Experiment 1: Test the hybrid algorithms without considering CPU capacity.	40
4.1.2	Experiment 2: Test the effect of considering the capacity of CPU.	43
4.1.3	Experiment 3: tested the effect of network delay with considering the Capacity of CPU	45
4.2	Summary	47
Chapter 5	Conclusion and Future Work	50
5.1	Conclusion.....	50
5.2	Future Work.....	51
References	52
Appendix A	1

List of Figures

Figure 1.1 Research Methodology	4
Figure 2.1 Cloud Computing Architecture	9
Figure 2.2 Types of Clouds	10
Figure 2.3 Load Balancing in Cloud Computing	13
Figure 2.4 Round Robin Algorithm.....	16
Figure 2.5 Random Algorithm	17
Figure 2.6 Equally Spread Current Execution Algorithm	18
Figure 2.7 Throttled Algorithm	19
Figure 2.8 Greedy Algorithm.....	20
Figure 2.9 Regions in Cloud Analyst Simulator	22
Figure 2.10 Users Based in CloudAnalyst Simulator	23
Figure 2.11 Data Center in CloudAnalyst.....	23
Figure 2.12 Data Center Details in CloudAnalyst	24
Figure 2.13 Hosts in Cloud Analyst Simulator	25
Figure 3.1 Steps to developing the proposed algorithm	33
Figure 3.2 The Proposed Hybrid Algorithm	34
Figure 3.3 Hybrid Algorithm Pseudo Code.....	37
Figure 4.1 All algorithm results Comparison without considering Capacity of CPU factor	42
Figure 4.2 All algorithm results Comparison for testing the effect Capacity of CPU factor	44
Figure 4.3 VM Allocations in Data center	45
Figure 4.4 All algorithm results Comparison for testing the effect of network delay	46

List of Tables

Table 4.1 Application development Configuration used in Experiment 1.....	41
Table 4.2 User bases configuration used in Experiment 1.....	41
Table 4.3 Data centers configuration used in Experiment 1.....	41
Table 4.4 Response Time and processing time results without considering Capacity of CPU factor	42
Table 4.5 Response time and processing time results for testing the effect of Capacity of CPU factor	43
Table 4.6 User Bases Configuration Used in Experiment3.....	45
Table 4.7 Response timer and processing time results for testing the effect of network delay.....	46

List of Abbreviations

DC	Data Center
ESCE	Equal Spread Current Execution
Exe	Execution Instruction
Mbps	Megabit per Second
Req	Request
RR	Round Robin
SLA	Service-Level Agreement
SOA	Service-Oriented Architecture
UB	User Base

Chapter 1

Introduction

Chapter 1 Introduction

In recent years, Cloud computing become a new computing model emerged from the rapidly development of internet. It leads the new IT revolution. Cloud computing considered as an evolution of distributed systems. Cloud computing is a heterogeneous environment offers a rapidly and on-demand wide range of services[1]. Heterogeneous environment means having different hardware characteristics including CPU, memory, storage and other hardware [2].The business owner can start and expand without invest in the infrastructure with lowering operating and maintenance cost. It has moved computing and data away from desktop and portable PCs, into large data centers[3]. It has the capability to harness the power of Internet and wide area network (WAN) to use resources that are available remotely, thereby providing cost effective solution to most of the real life requirements[3]. The National Institute of Standards and Technology's (NIST) define a Cloud computing as "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. "[4]

Load balancing is considered as one of the challenges in cloud computing, it is the major factor to improve the performance of the cloud computing. The current load balancing algorithms in cloud computing environment is not highly efficient [5]. Load balancing in cloud computing environment is very complex task till today, because prediction of user request arrivals on the server is not possible. Each virtual machine has different specification, so it becomes a very difficult to schedule job and balance the load among nodes [6].

Recently, we can find many research works that have been done on load balancing in cloud computing such as Round Robin, Equally Spread Current Execution and Throttled Load Balancing Algorithm. There is other works done using randomization such as ant colony algorithm [7].

In this research we propose a hybrid algorithm that takes advantages of both random and greedy algorithm. The experiments done using cloud analyst to test the performance of the proposed algorithm in heterogeneous of processors power. The experiments studied the effect of considering the capacity of CPU factor with the hybrid algorithm in heterogeneous environment of

hosts, and studied the effect of network delay on the hybrid algorithm. The results showed improvements on average response time and on processing time by considering the current resource information and the CPU capacity factor compared with other algorithms, and this means the performance has improved.

1.1 Statement of the problem

The current load balancing algorithms in heterogeneous of a processors power in cloud computing environment is not highly efficient[5], so our problem in this research is how to overcome this limitation by developing an efficient load balancing algorithm.

1.2 Objectives

1.2.1 Main objective

The main objective of this research is to propose a hybrid algorithm based on randomization and greedy algorithm to achieve efficient performance in heterogeneous of a processors power in cloud computing environment.

1.2.2 Specific objectives

The specific objectives of this research are:

- Design a new algorithm that adopts the characteristics of randomization and greedy to make an efficient load balancing and covers their disadvantages
- Simulate the proposed algorithm using Simulator. (e.g. CloudAnalyst)
- Evaluate the proposed algorithm system using response time metrics.
- Compare the results of proposed algorithm with those of other famous algorithms such as Round Robin and Equally Spread.

1.3 Research Methodology

In this research we propose a hybrid algorithm that takes advantages of both random and greedy algorithms. The algorithm considers the current resource information and the CPU capacity factor to achieve sufficient efficiency. Figure 1.1 shows the steps of the used methodology.

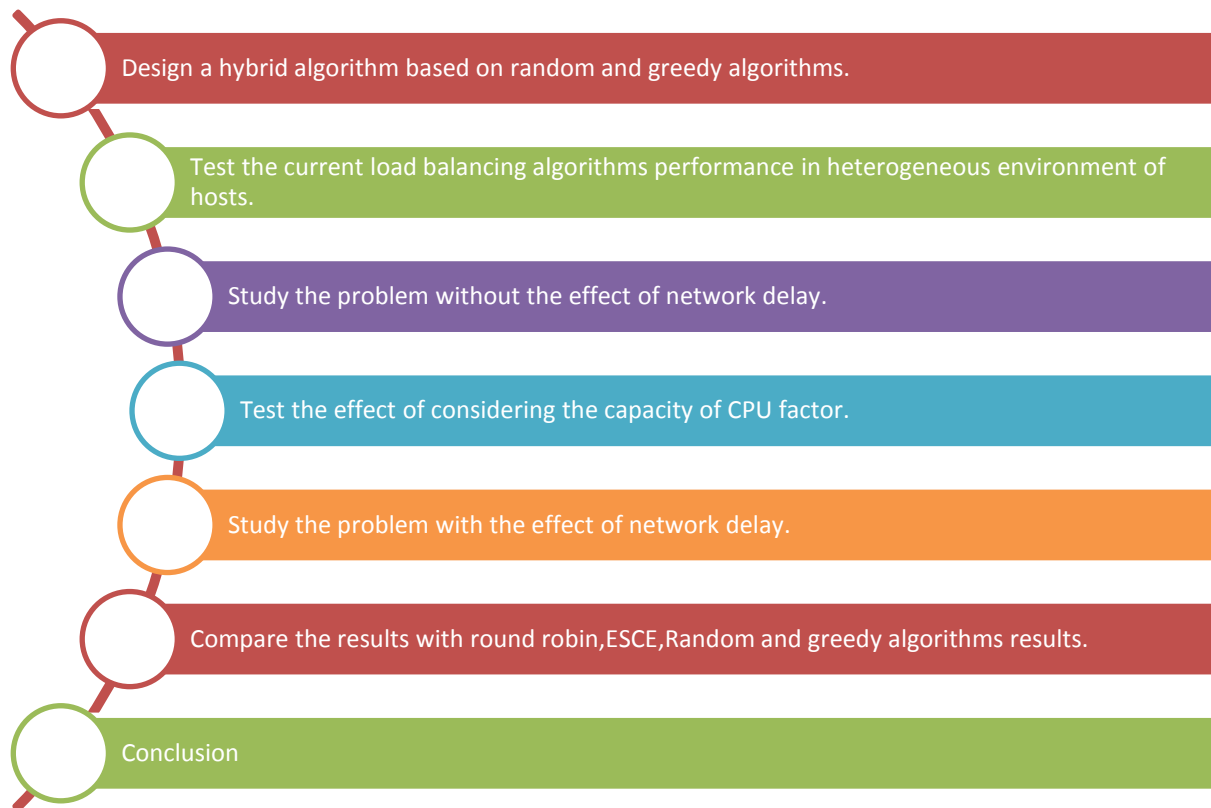


Figure 1.1 Research Methodology

First step we design a hybrid algorithm based on random and greedy algorithm, then testing the current load balancing algorithms performance in heterogeneous environment of hosts. Then studying the problem without the effect of network delay, and then testing the effect of considering the Capacity of CPU factor on the hybrid algorithm. Then testing the effect of network delay on the hybrid algorithm with considering the Capacity of CPU factor in a heterogeneous environment of hosts. Finally comparing the hybrid algorithm with Round robin, ESCE, Random and Greedy algorithms.

1.4 Significance of the thesis

- Improving the performance in heterogeneous of a processors power in cloud computing environment.
- Developing an efficient hybrid load balancing algorithm based on randomization and greedy algorithm.
- Studying the performance under different load balancing in low and high load.

- Studying the performance of load balancing in one data center with local users and with distributed users.

1.5 Scope and limitations

This research propose a hybrid load balancing algorithm which is mainly concentrate on overcoming the deficiencies in the performance of current algorithms. The limitations of this work are as follows:

- It considers the processor power factor; other specifications are out of scope.
- It focuses on normal arrival rate; other arrival rates are out of our scope.
- It is a dynamic non-distributed load balancing algorithm.
- It focuses only on improving scheduling performance in heterogeneous of a processors power in cloud computing environment.
- It is a local algorithm considers only one data center in one location.
- The performance of the proposed algorithm will be measured using simulator (cloud analyst), but not real experiments.

1.6 Research Organization

The research was organized as follows. Chapter 2 is devoted to literature review. In Chapter 3 we define the used model and the proposed algorithm. Chapter 4 is about experiments and results. Chapter 5 is for the conclusions and future directions.

Chapter 2

Technical Background and Related Works

Chapter 2 Technical Background and Related Works

This chapter defines cloud computing and its characteristics and models. We discuss cloud computing challenges and define the load balancing challenge and its types. We reflect a number of researches that are worked on enhancing load balancing.

2.1 Technical Background

Cloud computing become a new computing model emerged from the rapidly development of internet, it leads the new IT revolution. In this section we define the cloud computing and its characteristics and models, and we will define the virtualization and the benefits of virtualization in cloud computing. Also we discuss the cloud computing challenges and define load balancing challenge and its types.

2.1.1 Cloud Computing

In recent years, Cloud computing become a new computing model emerged from the rapidly development of internet, it leads the new IT revolution. Cloud computing considered an evolution of distributed systems, it is a heterogeneous environment offers a rapidly and on-demand wide range of services [1]. Heterogeneous environment means having different hardware characteristics including CPU, memory, storage and other hardware [2]. The business owner can start and expand without invest in the infrastructure with lowering operating and maintenance cost. It has moved computing and data away from desktop and portable PCs, into large data centers[3]. It has the capability to harness the power of Internet and wide area network (WAN) to use resources that are available remotely, thereby providing cost effective solution to most of the real life requirements[3]. Cloud computing is defined as a technical concept, where system users save the information on remote servers which are managed by others, and use the applications that are stored inside the server and executed from other locations, instead of from their own computers[8]. The National Institute of Standards and Technology's (NIST) define a Cloud computing as "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [4]. Large companies such as Google , Amazon and Microsoft provide more powerful, reliable and cost-efficient cloud platforms, Some examples of emerging Cloud computing are Microsoft Azure, Amazon EC2, Google App Engine [4, 9, 10].

The cloud is a virtualization of resources that maintains and manages itself [11]. It builds on a wide range of different computing technologies such as high-performance computing, distributed systems, virtualization, storage, networking, security, management and automation, Service-Oriented Architecture (SOA), Service-Level Agreement (SLA) and Quality of Service (QoS)[12].

2.1.2 Cloud computing features

Cloud computing provides several features that make it attractive to IT industry, such as :[4] [13].

- **No up-front investment:** The pricing model in cloud computing is based on a pay-per-use principle. This model gives them the ability to rent services and resources from cloud as he needs.
- **Lowering operating cost:** Cloud environment resources are allocated and de-allocated on demand and this can provide a considerable saving in operating costs since resources can be released when service demand is low.
- **Scalability and Elasticity:** the infrastructure providers have a large amount of resources and infrastructure. So they can easily expand its service to handle the growing service according to client demand [14]. On the other hand, Elasticity is the ability to scale resources both up and down when required. Allowing the dynamic integration and extraction of physical resources to the infrastructure. That's mean elasticity enables scalability [15].
- **Easy access:** the cloud services provided to users as a web-based services. So, they can access the services through any devices supported with Internet connections.
- **Reducing business risks and maintenance expenses:** Shifts the business risks such as hardware failures to infrastructure providers, because providers have better expertise and resources to manage these risks [14].
- **Virtualization:** Virtualization hides a computing platform's physical characteristics from users [16] [17], It allows abstraction and isolation of lower level functionalities and underlying hardware.
- **Mobility:** Cloud Computing means mobility because users can access applications [18] through internet easily at any point of time.

2.1.3 Cloud Service Model

Cloud Computing can be delivered through such delivery models as follow.

- **Infrastructure as a Service (IaaS):**

This model of Cloud computing provide Hardware as a Service via Internet such as storage, CPU and other. There are many IaaS providers such as Amazon Elastic Cloud Compute (EC2), Rackspace[8] [13].

- **Platform as a Service (PaaS):**

Cloud computing provide a platform as a services that required for building application, where user using tools and libraries for Cloud service providers, and also consumers deployed their applications without costing of hardware where providers of services provide the network, storage. There are many PaaS providers such as Google App Engine, Windows Azure[8] [13].

- **Software as a Service (SaaS):**

Focus on providing different software hosted on the Cloud and usually referred to as on-demand-software, where in this type of service, consumer will have to pay for usage of software. Usually consumer access to the software via the Internet, therefore, user uses the software don't need any integration with other system[8] [13]. There are many SaaS provider such as Google Apps, Salesforce.com as shown in Figure 2.1 [4].

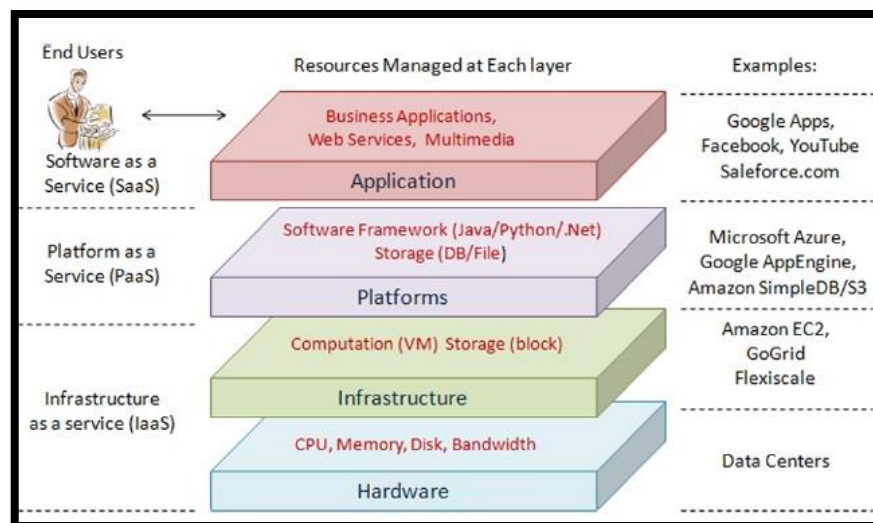


Figure 2.1 Cloud Computing Architecture

2.1.4 Cloud Deployment Model

There are different types of clouds as shown in Figure 2.2 [19], each with its own benefits and drawbacks.

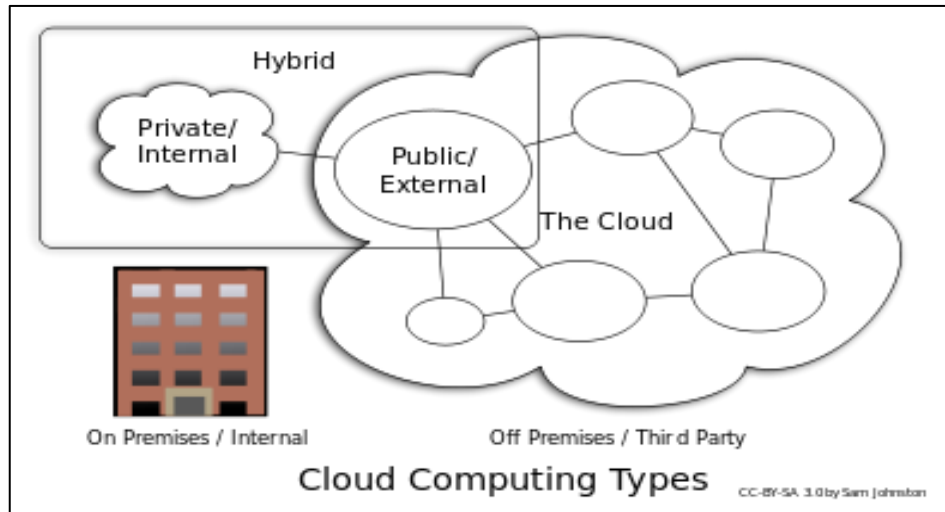


Figure 2.2 Types of Clouds

- **Public clouds:** A cloud in which service providers offer their resources as services to the general public. Public clouds offer several key benefits to service providers, including no initial capital investment on infrastructure and shifting of risks to infrastructure providers. However, public clouds lack fine-grained control over data, network and security settings, which hampers their effectiveness in many business scenarios[4] [15].
- **Private clouds:** Also known as internal clouds, private clouds are designed for exclusive use by a single organization. A private cloud may be built and managed by the organization or by external providers. A private cloud offers the highest degree of control over performance, reliability and security. However, they are often criticized for being similar to traditional proprietary server farms and do not provide benefits such as no up-front capital costs[4] [15].
- **Hybrid clouds:** A hybrid cloud is a combination of public and private cloud models that tries to address the limitations of each approach. In a hybrid cloud, part of the service infrastructure runs in private clouds while the remaining part runs in public clouds[4] [15].

2.1.5 Virtualization

Virtualization separates resources and services from the underlying physical delivery environment [13]. Virtualization is considered as a core of cloud computing technologies and one of the most important technologies that enabled this paradigm [16] [17]. Virtualization hides a computing platform's physical characteristics from users [16] [17]. It allows abstraction and isolation of lower level functionalities and underlying hardware. This enables portability of higher level functions and sharing and/or aggregation of the physical resources [20].

Virtualization means "something which isn't real", but gives all the facilities of a real [6]. It is the software implementation of a computer which will execute different programs like a real machine [21].

Virtualization has three characteristics that make it very related with cloud computing which are [13]:

1- Partitioning:

By partitioning the available resources, many applications and operating systems can run in a single physical system.

2- Isolation:

By isolation, each virtual machine can run in its host with others virtual machine without effect on others. So, if one virtual instance failed, it doesn't affect the other virtual machines.

3- Encapsulation:

A virtual machine encapsulated and stored as a single file, so a virtual machine can be presented to an application as a complete entity without interfere with another application.

2.1.6 Cloud Computing Issues and Challenges

There are many issues and challenges emerged from cloud computing and are required to be addressed properly as following:

- **Security**

It is clear that the security issue has played the most important role in Cloud computing. Security issues such as data loss, phishing, privacy and other threats, Whether at the enterprise level or individual level that use the pooled computing resources in cloud computing, has introduced new

security challenges. So, we need novel techniques to reduce the impact of the endless dangers in the cloud computing environment [22].

- **Performance**

It is the second issue in cloud computing [22]. Poor performance can be caused by lack of resources such as disk space, limited bandwidth, lower CPU speed, memory, network connections etc. The data intensive applications are more challenging to provide proper resources. Poor performance can result in end of service delivery, loss of customers and reduce revenues [15].

Performance can be based on different methods, tools and simulations for cloud environments such as fuzzy systems and a tool like Cloud Analyst [23]. There is a series of factor that affect the performance such as:

- Security.
- Recovery and Fault tolerance.
- Service level agreements.
- Bandwidth.
- Storage capacity.
- Physical memory.
- Disk capacity.
- Processor Power.
- Availability.
- Number of users and Workload.
- Usability.
- Scalability.
- Location, data centers and their distance from a user's location.

And there is a series of criteria for evaluating the performance such as [23] [24]:

- Average response time per unit time.
- Average waiting time per unit time.
- Workload to be serviced per second (Mbps) or a unit of time.
- Throughput (Req / Sec).
- The average time of processing (exe / sec).
- Percentage of CPU utilization.

- The number of requests executed per unit time.
- The number of requests per unit time buffer.
- The number of rejected requests per unit time.

- **Load balancing**

Load balancing is a process of reassigning the total load to the individual nodes of the collective system to improve both resource utilization and job response time. It also avoids a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all nodes in the system approximately equal amount of work at any instance of time [3, 25]. The objective of load balance is to achieve optimal resource utilization, maximize throughput, minimum response time, and avoid overload [18]. The heterogeneous environment considered as a major concern [26-28] because the heterogeneous environment consist of heterogeneous resource, so the behaves of heterogeneous cloud different and has different attributes and different response times for any process [27, 29].

Although load balancing in cloud computing is based on standard load balancing, it differs from classical load-balancing such as in parallel computing. In cloud computing the architecture and implementation of the load balancing process is different according to the use of commodity servers to perform the load balancing, which provides for new opportunities and economies of scale [30]. Figure 2.3 presents load balancing in cloud computing [31].

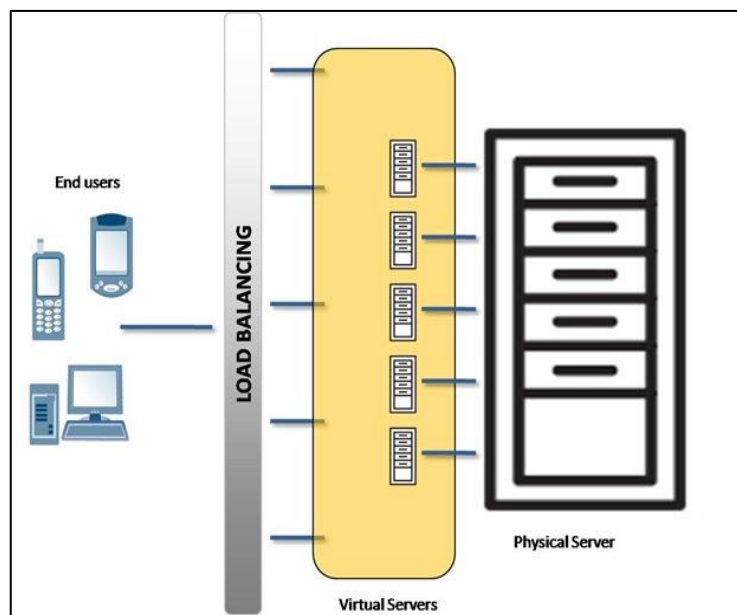


Figure 2.3 Load Balancing in Cloud Computing

As seen in Figure 2.3, the data center required a load balancing policy to process the users requests. The load balancer responsible to assigns the virtual machine to the user request. Then the data center sends the response to the users after processing the Request.

The load balancing is very important in cloud computing environment. The major goals of load balancing algorithms are:

- Achieve an overall improvement in system performance at a reasonable cost [18].
- To have a backup plan in case the system fails even partially [21].
- To accommodate future modification in the system: the distributed system can change such as applying new topology and scale up. So a load balancing algorithm must be scalable and flexible to handle this changes [18].

The mathematical model of load balancing is defined as follows:

Let say that there are n set of Load or requests need to be scheduled given as:

$$L = \{L_1, L_2, \dots, \dots, L_n\} \quad (1)$$

And there are K set of Virtual Machines in a Datacenter given as:

$$V = \{V_1, V_2, \dots, \dots, V_k\} \quad (2)$$

The current Datacenter load is given as:

$$DL = \{VL_1, VL_2, \dots, \dots, VL_k\} \quad (3)$$

We need to find a function $f(L)$, where the set of load L can be mapped to the set of Virtual Machines V , making the Load VL_i of each Virtual Machine V_i be essentially equal, that is:

$$VL_1 \approx VL_2 \approx \dots \approx VL_k \quad (4)$$

Let us use τ_o to reflect the time needed for executing task L_o on the Virtual Machine V_i , so the time needed for executing all the tasks on the Virtual Machine V_i is as follows:

$$t_i = \sum_{o \in f(L_i)(i=1, \dots, n)} \tau_o \quad (5)$$

When $k = 1$, that means there is only one Virtual Machine, and all the tasks should be executed serially on this Virtual Machine, so the time needed for execution is the sum of all the time, which can be represented as T_1 shown below:

$$T_1 = \sum \tau_o \quad (o = 1, \dots, n) \quad (6)$$

When $K > 1$, that means there is more than Virtual Machine, and the tasks can be shared to multiple server nodes for dealing with in parallel, the time needed is represented as T_k shown below:

$$T_k = \max_{i=1, \dots, n} t_i \quad (7)$$

Thus, the goal of load balancing is to solve the function $f(L)$ to get the minimum of T_k in case of $VL_1 \approx VL_2 \approx \dots \approx VL_k$

Load balancing algorithm can be divided into two categories as 1) Static and 2) Dynamic. [32] [25].

- **Static algorithms:**

Static algorithms divide the traffic equivalently between servers and the load balancing strategy has made by load balancing algorithm at compile time [6] By this approach the traffic on the servers will be disdained easily and consequently it will make the situation more imperfectly. A general disadvantage of all static schemes is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load. Round robin algorithms are a static load balancing algorithm because the work load distributions between processors are equal [24].

- **Dynamic algorithms:**

In dynamic algorithms decisions on load balancing are based on current state of the system. No prior knowledge is needed for load balancing. So it is better than static approach. Dynamic load balancing can be done in two ways [7] [24]:

o **Distributed dynamic load balancing:**

In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them. A benefit, of this is that even if one or more nodes in the system fail, it will not cause the total load balancing process to halt; it instead would affect the system performance to some extent.

o **Non-distributed dynamic load balancing:**

In the non-distributed there is one node responsible for load balancing of the whole system. The other nodes interact merely with the central node [32]. In this research the proposed algorithm will be a type of non-distributed dynamic load balancing.

- **Existing Load Balancing Algorithms**

This section presents some of the popular load balancing algorithms which are used in cloud computing environment. In our work, we are going to make experiments on some of these algorithms and compare them with our work.

1. Round Robin Algorithm

It is considered as the most basic and the least complex scheduling algorithm [33], it use the concept of time quantum and each processor take a time quantum, the processes are divided between all processors as seen in Figure 2.4 [25]. Each process is assigned to the processor in a round form order. If the process does not complete in a given time, it will be placed at the end of waiting queue, The drawback of this algorithm is at any point of time some nodes may be heavily loaded and others remain idle [18, 25].

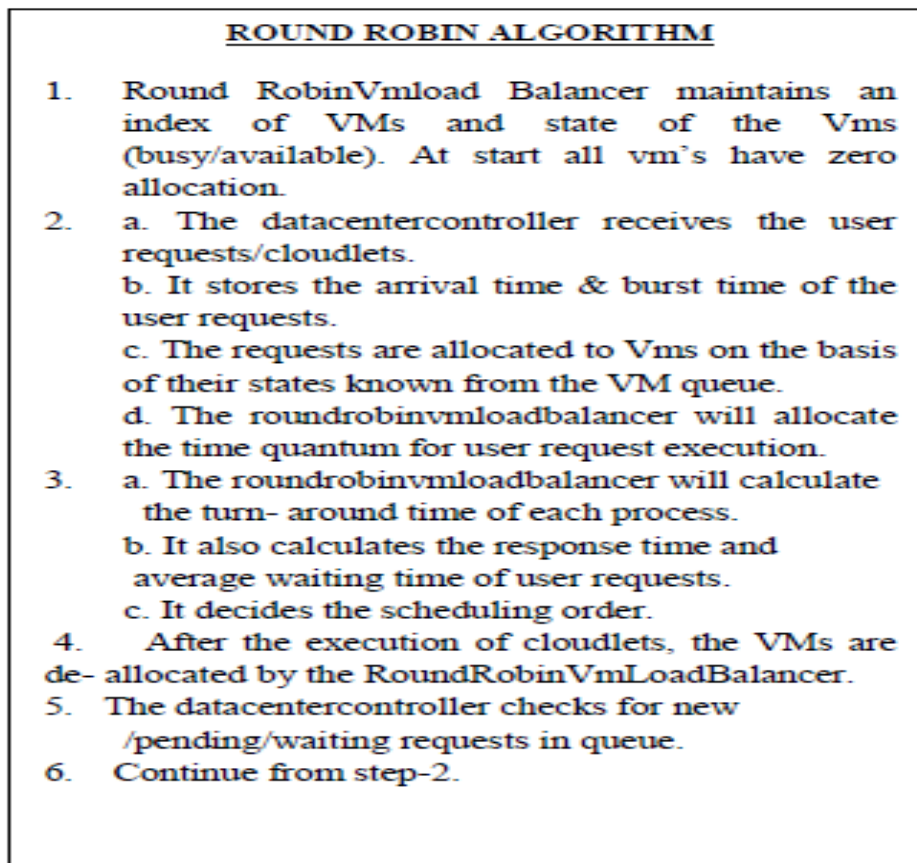


Figure 2.4 Round Robin Algorithm

2. Random Algorithm

The idea of random algorithm is to randomly assign the selected jobs to the available Virtual Machines (VM) [24]. As seen in Figure 2.5 the algorithm does not take into considerations the status of the VM, which will either be under heavy or low load. Hence, this may result in the selection of a VM under heavy load and the job requires a long waiting time before service is obtained. The complexity of this algorithm is quite low as it does not need any overhead or preprocessing [18, 34].

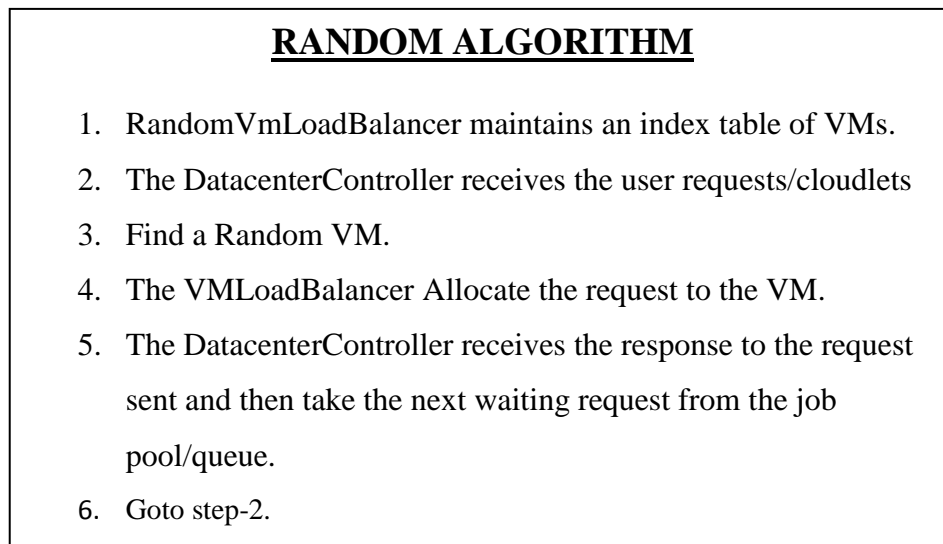


Figure 2.5 Random Algorithm

3. Equally Spread Current Execution Algorithm.

Equally spread current execution algorithm as shown in Figure 2.6 [25]. it distribute the load randomly by checking the size and transfer the load to that virtual machine which is lightly loaded or handle that task easy and take less time , and give maximize throughput. It is spread spectrum technique in which the load balancer spread the load of the job in hand into multiple virtual machines [25, 34].

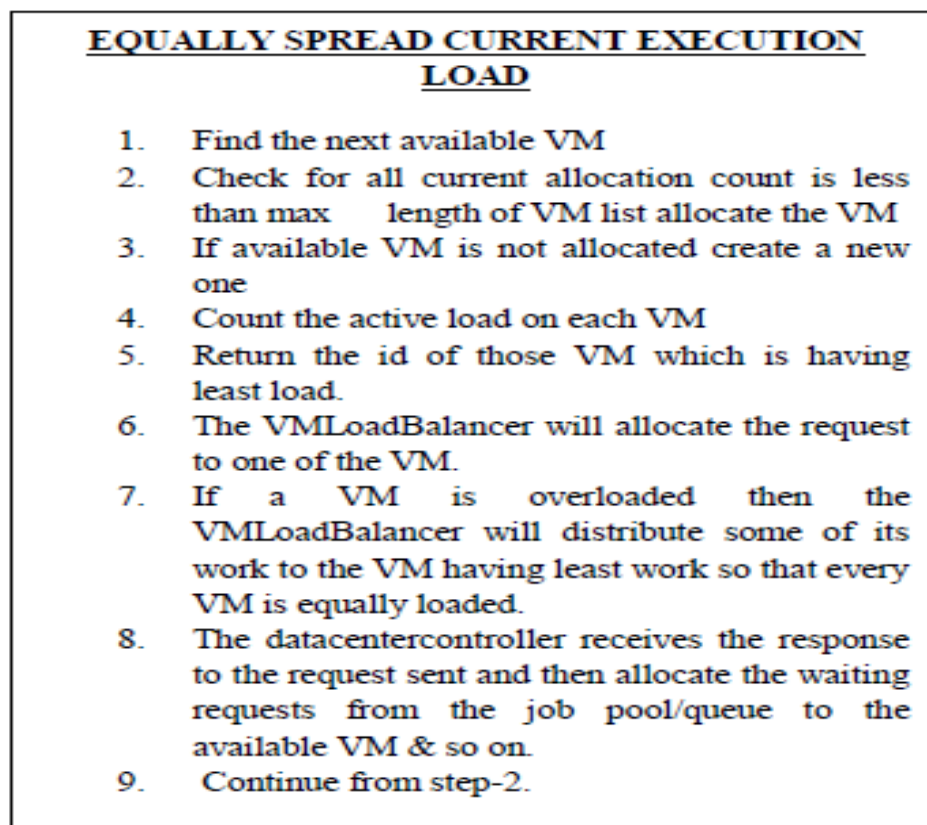


Figure 2.6 Equally Spread Current Execution Algorithm

4. Throttled Load Balancing Algorithm:

In this algorithm the load balancer maintains an index table of virtual machines as well as their states (Available or Busy) [3, 33]. As seen in Figure 2.7 [25] the data center queries the load balancer for allocation of the VM. The load balancer scans the index table from top until the first available VM is found or the index table is scanned fully. If the VM is found, the data center assigns the task to the VM by id, but if VM is not found, the load balancer returns -1 to the data center. Then the data center will put this job in a queue [25].

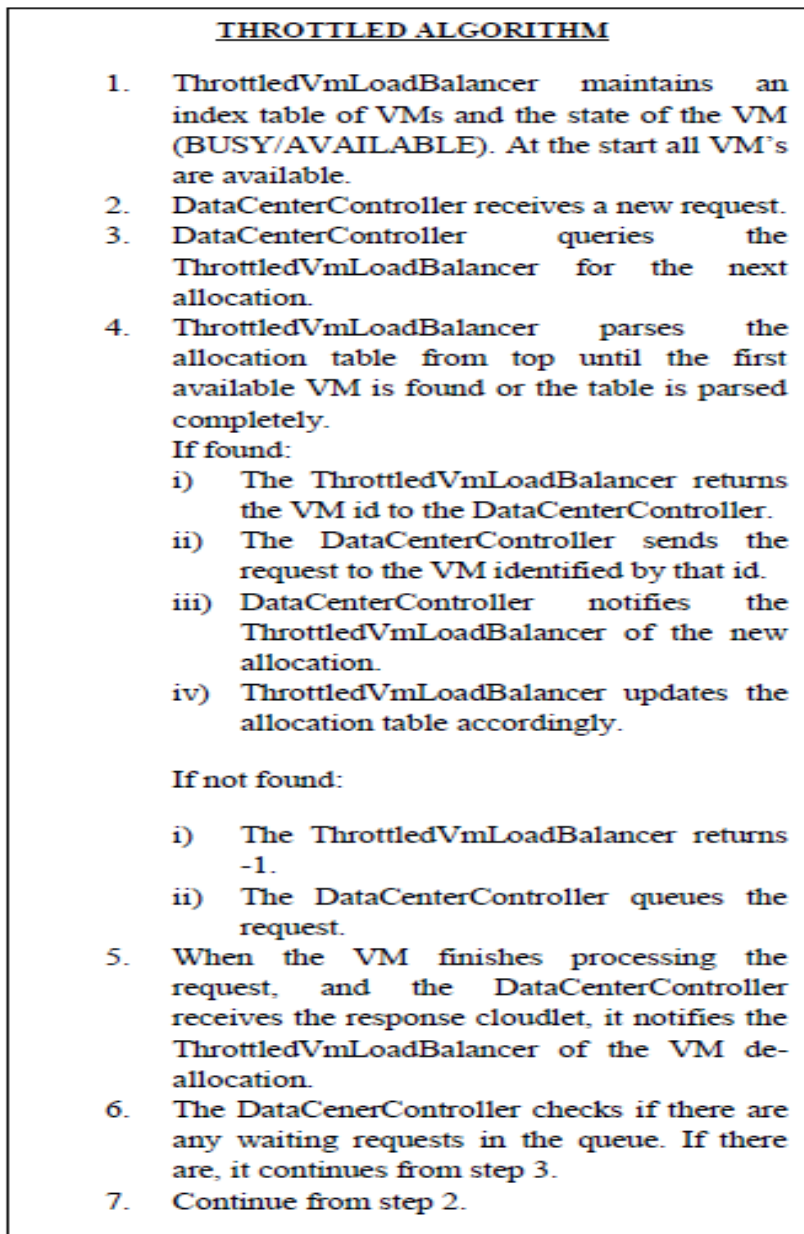


Figure 2.7 Throttled Algorithm

5. Greedy Algorithm:

A greedy algorithm as shown in Figure 2.8 always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution [35]. It always selects the best site to execute the job according to specific criteria such as: shortest queue length, least work load, and least queuing time.

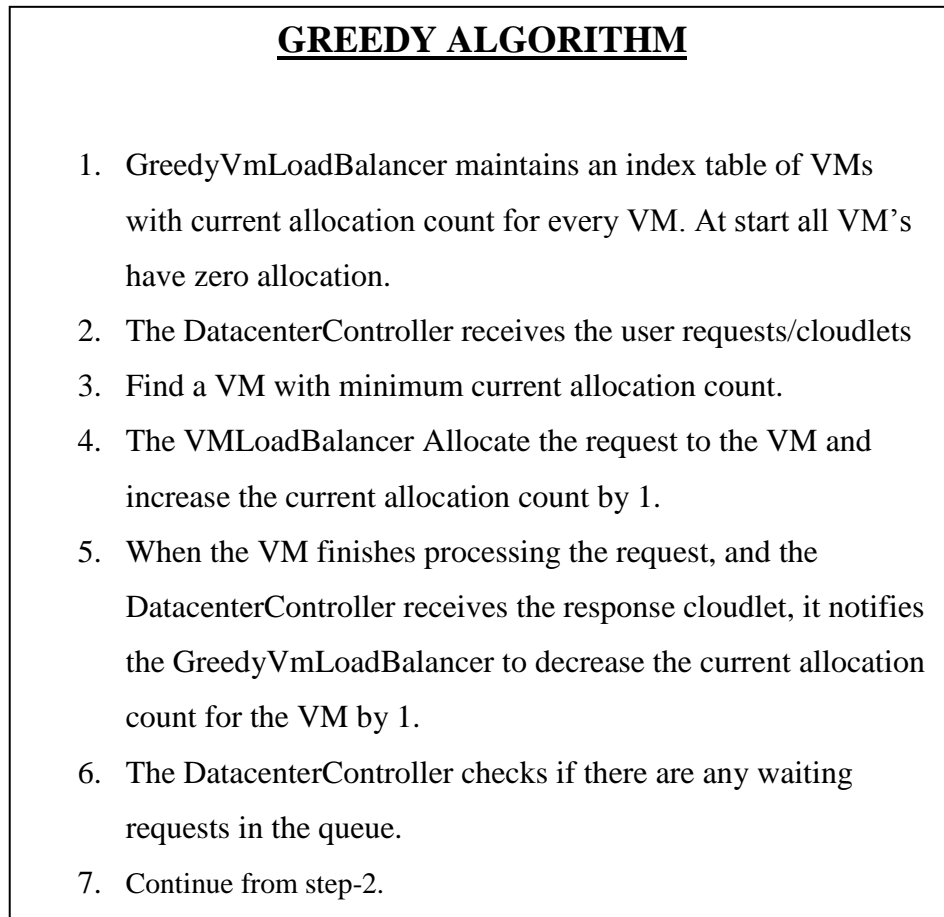


Figure 2.8 Greedy Algorithm

6. Minimum Completion Time Algorithm:

Minimum Completion Time algorithm is an example of greedy load balancing algorithm. The main idea of this algorithm [34] is to assign the received task to the available VM that can offer the minimum completion time taking into account its current load. The completion time for every VM is calculated depending on the processor speed and the current load on the VM. When a request arrives, the load balancer scans the available VMs in order to determine the most appropriate machine which has the minimum completion time, to perform the task. Although greedy algorithm selects the best VM to handle the received task, the selection processes needs some complex computation to find the best VM.

In this research we will propose a hybrid algorithm that takes advantage of both random and greedy algorithms; every algorithm has some advantages and some limitations. For example, the random algorithm which randomly selects a VM to process the received tasks, does not need complex computation to make a decision but it does not select the best VM. On the other hand the greedy algorithm selects the best VM to handle the received task, but the selection process needs some complex computation to find the best VM.

2.1.7 Cloudsim

The proposed algorithm is tested in a cloud computing environment. We have two choices to test it, the first choice is to use a real test such as Amazon EC2, and the second is to use simulation tools to simulate a cloud environment. In our work we prefer to use a simulator, because using real test limits the experiments to the scale of the infrastructure, and makes the reproduction of results an extremely difficult undertaking [10]. Also it is very difficult and time consuming to measure performance in real cloud environment [18]. In addition, accessing to the real infrastructure incurs payments in real currency.

The simulation framework has some features as follow[36] :

1. Support for modeling and instantiation of large scale Cloud computing infrastructure, including data centers, virtual machines , service brokers, scheduling, and allocation policies.
2. Support for virtualization, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node.
3. Flexibility to switch between space-shared and time-shared allocation.

Cloud Analyst is a graphical simulation tool based on Cloudsim for modeling and analysing behavior of cloud computing environments, which supports visual modeling and simulation of large-scale applications that are deployed on cloud Infrastructures [37].

The main features of Cloud Analyst are as following [38]:

1. Easy to use Graphical User Interface (GUI).
2. Ability to define a simulation with a high degree of configurability and flexibility. Simulation of complex systems such as Internet applications depends on many parameters.

3. Repeatability of experiments.
4. Graphical output.
5. Ease of extension.

The cloud analyst allows setting location of users, number of user and number of request per user per hour. And also it allows setting the location of the data centers, number of virtual machines, number of processors, amount of storage, network bandwidth and other necessary parameters [25].

The main components of Cloud-Analyst are as follow:

1. Region

The world is divided into 6 regions based on the 6 main continents in the world. The other main entities such as user bases and data centers belong to one of these regions [39]: (N-American, S-American, Europe, Asia, Africa and Oceania). As shown in figure 2.9.

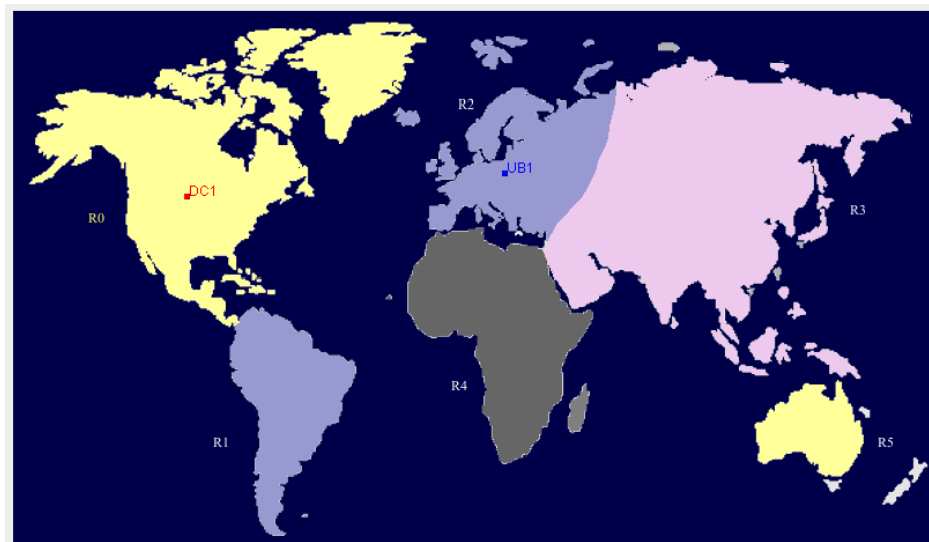
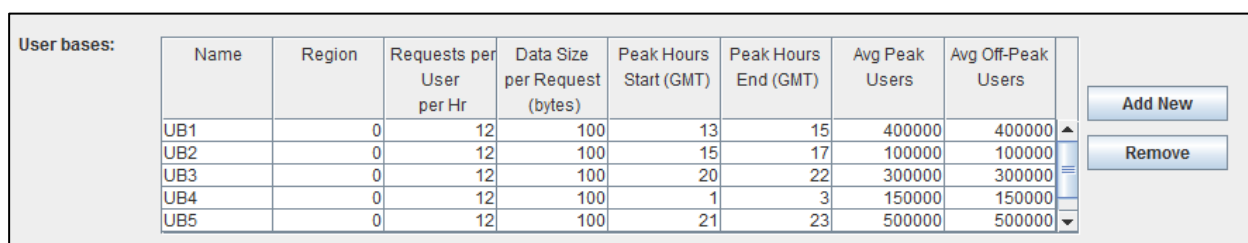


Figure 2.9 Regions in Cloud Analyst Simulator

2. Users based

A user base models a group of users that is considered as a single unit in the simulation and its main responsibility is to generate traffic for the simulation. A single user base may represent thousands of users but is configured as a single unit and the traffic generated in simultaneous bursts representative of the size of the user base. The modeler may choose to use a user base to represent a single user, but ideally a user base should be used to represent a larger number of users for the efficiency of simulation [40]. Figure 2.10 shows the user base configuration.

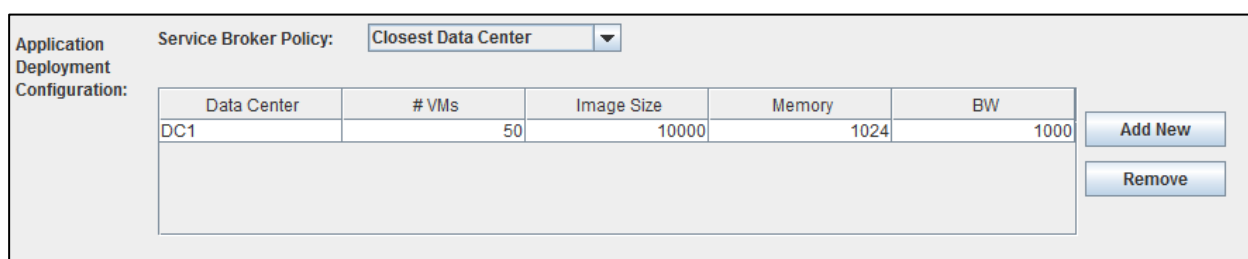


Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	0	12	100	13	15	400000	400000
UB2	0	12	100	15	17	100000	100000
UB3	0	12	100	20	22	300000	300000
UB4	0	12	100	1	3	150000	150000
UB5	0	12	100	21	23	500000	500000

Figure 2.10 Users Based in CloudAnalyst Simulator

3. Datacenter

This component is used to control the various data center activities [41] such as VM creation and destruction and does the routing of user requests received from user bases via the Internet to the VMs [39]. It encapsulates a set of compute hosts (servers) that can be either homogeneous or heterogeneous as regards to their resource configurations [36]. Figure 2.11 and Figure 2.12 shows the data center configuration.



Data Center	# VMs	Image Size	Memory	BW
DC1	50	10000	1024	1000

Figure 2.11 Data Center in CloudAnalyst

Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM \$/Hr	Memory Cost \$/s	Storage Cost \$/s	Data Transfer Cost \$/Gb	Physical HW Units
DC1		0x86	Linux	Xen	0.1	0.05	0.1	0.1	5

Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	2000	TIME_SHARED
1	204800	100000000	1000000	5	5000	TIME_SHARED
2	204800	100000000	1000000	2	9000	TIME_SHARED
3	204800	100000000	1000000	2	10000	TIME_SHARED
4	204800	100000000	1000000	2	15000	TIME_SHARED

Figure 2.12 Data Center Details in CloudAnalyst

4. ServiceBroker

The responsibility of this component is to model the service brokers that handle traffic routing between user bases and data centers. The service broker can use one of the routing policies from the given three policies which are closest data center, optimize response time and reconfigure dynamically with load [25].

5. Hosts

This class models a physical service in a cloud-based data center. It contains an amount of memory and storage, a list of processing elements (to represent a multi-core machine), an allocation policy for sharing the processing power among virtual machines, and policies to provisioning memory and bandwidth to the virtual machines [42]. Figure 2.13 shows the hosts configuration.

Physical Hardware Details of Data Center : DC1						
Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	2000	TIME_SHARED
1	204800	100000000	1000000	5	5000	TIME_SHARED
2	204800	100000000	1000000	2	9000	TIME_SHARED
3	204800	100000000	1000000	2	10000	TIME_SHARED
4	204800	100000000	1000000	2	15000	TIME_SHARED

Figure 2.13 Hosts in Cloud Analyst Simulator

6. VmLoadBalancer

VM Load balancer is useful to determine which VM should be assigned the requests (Cloudlet) for processing. Three policies are included currently in the Cloud-analyst which are Round-robin Load Balancer, Active Monitoring Load Balancer and Throttled Load Balancer [40, 41]

7. VMProvisioner

This abstract class represents the provisioning policy that a VM monitor utilizes for allocating VMs to Hosts. The main responsibility of the VMProvisioner is to select available host in a data center, which meets the memory, storage, and availability requirement for a VM deployment [10].

8. Virtual Machine:

This class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component. The Host can simultaneously instantiate multiple VMs and allocate cores based on predefined processor sharing policies (space-shared, time-shared) [10].

9. Cloudlet

Cloudlet is a grouping of user requests. The number of requests bundled into a single Cloudlet is configurable in CloudAnalyst. The Cloudlet carries information such as the size of a request execution command, size of input and output files, the originator and target application used for routing by the Internet and the number of requests [43].

The CloudAnalyst simulation has some metrics as follows [23]:

- Overall response time: Minimum, maximum and average.
- Overall processing time in the data center: Minimum, maximum and average.
- Response time per user: Minimum, maximum and average.
- Minimum, maximum and average time per data center.
- Virtual machine total cost.
- Cost per VM of Data Center.
- Cost of data in each data center.
- Total cost in each data center.

The Routing of user requests in Cloud-Analyst is done in nine steps as follows [39]:

1. User base generates an Internet Cloudlet, with application id for application and also includes name of the user base itself as originator for routing back the response.
2. Request is sent to the Internet with zero delay.
3. Internet consults the service broker for the data center selection. The service broker uses any one of the service broker policy based on the Request.
4. Service broker sends information about selected data center controller to the Internet.
5. Internet adds appropriate network delay with the Request and sends to the selected data center controller.
6. Selected data center controller uses any one of the virtual machines load balancing policy.
7. Virtual machines load balancer assigns the virtual machine to the user request.
8. Selected data center sends the response to the Internet after processing the Request.
9. Internet uses the originator field of the Cloudlet information and adds appropriate network delay with response and sends it to the user base.

2.2 Related works

Many researchers' proposed different algorithms in load balancing and job scheduling in cloud computing. In this section we review a number of researches that worked on enhancement of load balancing.

James et al in [44] provide an algorithm called weighted Active VMLoadBalancer to decrease the response time, and data processing time. In this algorithm they compute the power of VM's in the datacenter and use index table to store the count of requests that are currently allocated in VM. When a new request is received, the load balancer looks at the table and identifies the least VM loaded. Then the result return to the datacenter and the datacenter allocates the VM. Finally when the VM finishes, it will notify the datacenter and the datacenter will de-allocate it.

The authors built index table to monitor each node in the system to quickly know the status of the node and to allocate the best VM. Although this algorithm decrease the response time, but they need to compares their results with other algorithms such as ESCE and Throttled in order to evaluate the results.

Sethi et al in [30] introduce a load balancing algorithm using fuzzy logic with Round Robin (RR) algorithm. The algorithm is based on various parameters such as processor speed, and assigned load in VM and etc. The algorithm maintains the information of each VM and numbers of requests currently allocated to VM. When a new request is received, the load balancer searches for the least loaded VM and allocate it, but if there are more than one VM, the selection will be based on processor speed and load in VM using fuzzy logic.

This algorithm enhanced the performance of load balancer and decreased the response time. In addition, the results referred that its performance is better than RR algorithm. The drawback of this approach that authors had focused only on how to decrease the response time of job scheduling and they ignored talk about processing cost. In addition, the researchers compared their results with only RR algorithm which had been enhanced and improved by many researchers before.

Sharma et al in [3] propose a new enhancement scheduling algorithm EEA, the main purpose of the algorithm is to find the expected response time of each VM to achieve the maximum throughput and decrease response time to avoid overhead. They compared their algorithm with three algorithms, Round Robin (RR), Equal Spread Current Execution scheduling and Throttled algorithm.

The result shows that overall response time and data center processing time is improved as well as cost is reduced in comparison to the existing scheduling parameters.

However this approach is limited on enhancement of response time and how to achieve the maximum throughput but does not handle the fault tolerance and the problem caused by deadlocks and server overflow.

Hu et al in [45] propose a new algorithm to enhance job scheduling using a genetic information. The algorithm uses a historical data and current state of the system. It makes a mapping relationship between the set of physical machines and the set of VMs. It chooses the least-affective solution by computing ahead influence of the system after the deployment of the needed VM resources. They used some equation to find the best scheduling solution using population.

The experimentation results show an improvement in the utilization of resources. On the other hand, the proposed algorithm has high cost to store and retrieve the historical data of the system nodes, and this may also increase the response time and the processing cost.

Fang et al in [46] try to obtain high resource utilization and meet dynamic requirements of task by providing a two level task scheduling mechanism based on load balancing in cloud computing. They paper improve the response time, resources utilization by mapping task to VMs and then VMs to host resources. They use the first level of scheduling (from user's application to the VM) to create a description of VM including the task of computing resources, network resources, storage resources, etc. and used the second level scheduling (from the VM to host resources) to find appropriate resource for VM.

This approach may have succeeded in improving the resource utilization, but we think that using two levels of task scheduling would increase the response time compared with other load balancing algorithms.

Sharma et al in [47] proposed a new algorithm to enhance response time of each VM. The proposed algorithm collects information about all VMs in a list and uses it to allocate appropriate VM where status is available. When a new request is received, the load balancer looks at the table and identifies VM whose current allocation count is less than max allocation, and then check its status. The result is returned to the datacenter and then the data center allocates this resource to the request. When the VM is finished, it notifies the datacenter to de-allocate it. The drawback of

this algorithm is in some case such as the high workload it may increase the waiting queue because the allocation depends on the available status only.

Subramanian et al in [48] propose a new algorithm that combine the advantages of three algorithms and overcomes their disadvantages. These three algorithms were: greedy, round robin, and power saver algorithm. The algorithm focused on best utilization of resources and minimizing the power consumption. It scheduled the VMs to the nodes depending on their priority value, which varies dynamically based on their load factor. When a request is received, the node with the maximum available resource is determined and then it is checked whether the node had a load factor less than 80%. If the highest priority node had a load factor less than 80%, then the VM is scheduled to that node, otherwise it checks the next maximum resource. The idle nodes (which, no VM is allocated to them) are turned off to save power. The main drawback of this algorithm is in some case such as the high workload, the power saver algorithm will be inactive because all the VM would be busy in most of the processing time and this would affect the performance.

Mishra et al in [7] propose a new algorithm that depend on ant colony technic. Ants depend on the strength of the ant's pheromone to select the optimal path that leads to their destination. In the same way each node in the network has a pheromone. Each row in the pheromone table represents the routing preference for each destination, and each column represents the probability of choosing a neighbor as the next hop. If an ant is at a choice point when there is no pheromone, it makes a random decision. If the pheromone exists, the node with high probability is selected and then the pheromone table is updated by increasing the probability of this node and decreasing other nodes probabilities. The main drawback of this algorithm is that it does not consider the current workload information for each node. So in some case there are some nodes may be heavily loaded and others remain idle.

Singh et al in [49] develop a new heterogeneous load balancing algorithm to distribute the load across a number of servers. They create VMs of different datacenters according to host specification including core processor, processing speed, memory, storage etc. Then allocate weight count according to the RAM allocated to the VMs in the datacenter. They use a data structure to maintain weight count and the current allocation count of the VM. They allocate the VM which have available status and have a higher RAM. When allocating a new VM, the algorithm returns the VM id to the DataCenterController, and then updates the allocation count for that VM

and adding the new allocation to the busy list. When the VM finishes processing the request the algorithm de-allocates the VM and removes the VM from the busy list.

The main drawback of the algorithm is the authors allocates the VM which have higher RAM specification, but they ignores others specification such as processor power. On other hands they do not present any results and comparison with other algorithms.

Dave and Maheta in [33] propose new load balancing algorithm based on round robin algorithm, they made a modification on round robin algorithm by implementing a dynamic time Quantum based on algorithm execution round. The result shows an improvement in response time as compared to normal round robin algorithm. The drawback of this paper is that authors had focused only on how to decrease the response time and they ignored talking about processing cost. In addition, they need to compares their results with other algorithms such as ESCE and Throttled in order to evaluate the results.

Singhal and Jain in [50] propose a load balancing algorithm using Fuzzy Logic, the algorithm focuses on a public cloud. The main idea of the algorithm is partitioning the cloud to several partitions and each partition having its own load balancer, and there is a main controller which manages all these partitions. With the idle partition status they use a fuzzy logic and in the normal partition status they use a global swarm optimization based load balancing strategy. The result shows enhancements in resource utilization and availability in cloud computing environment. The drawback of this approach is the difficulty of testing the technique in a real environment to make sure that it has achieved good results.

Recently, we can find other research works done on load balancing in cloud computing using randomization such as ant colony optimization.

Zhan and Huo in [7] provide a mixed algorithm between Particle Swarm Optimization (PSO) and Simulated Annealing (SA) algorithms to benefit from the characteristics of the strong randomization of PSO algorithm.

2.3 Summary

In this chapter we defined the cloud computing and its characteristics and models, and we defined the virtualization and the benefits of virtualization in cloud computing. We discussed on the cloud computing challenges. We defined the load balancing challenge and its types. We also defined the cloudAnalyst simulator and their components.

As presented in the related works we can conclude that the current load balancing scheduling algorithms in cloud computing environment have some deficiency and this would affect the performance. So we need to overcome this limitation by developing an efficient load balancing algorithm that consider the response time in order to improve the performance of heterogeneous of a processors power in cloud computing system.

Chapter 3

Proposed Algorithm

Chapter 3 Proposed Algorithm

The current load balance scheduling algorithms in heterogeneous of a processors power in cloud computing environment is not highly efficient. The main objective of this research is to achieve efficient performance in heterogeneous of a processors power in cloud computing environment. In this chapter we will present the proposed a hybrid algorithm that takes advantages of both random and greedy algorithms.

3.1 Proposed Algorithm

In this research we proposed a hybrid algorithm that takes advantages of both random and greedy algorithms. The random algorithm which randomly select a VM to process the received tasks, does not need complex computation to make a decision but it does not select the best VM. On the other hand greedy algorithm selects the best VM to handle the received task, but the selection process needs some complex computation to find the best VM. The steps that followed to accomplish this work presented in figure 3.1

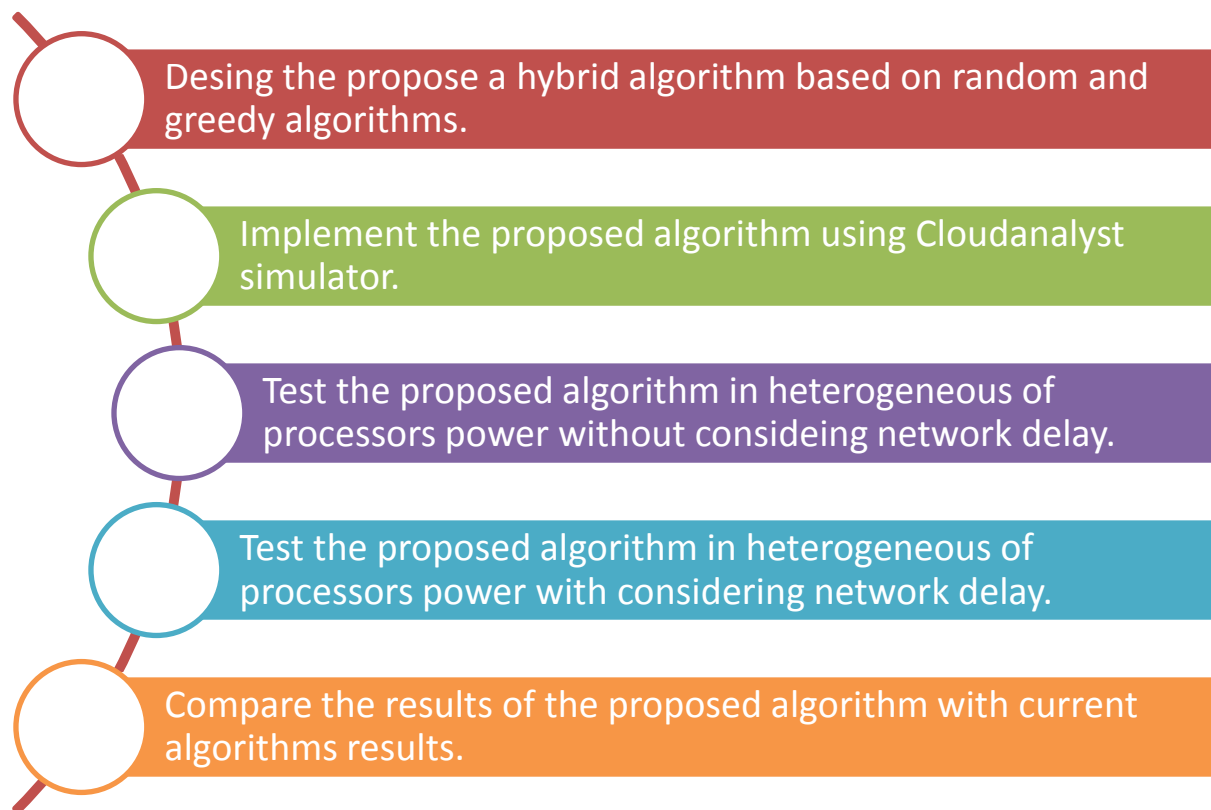


Figure 3.1 Steps to developing the proposed algorithm

First we design the proposed a hybrid algorithm based on random and greedy algorithms. The design process includes development of the model, specification and designing the algorithm, checking the correctness of Algorithm, and analysis of Algorithm. Then we implement the proposed algorithm using Cloudanalyst simulator. After that we test the proposed algorithm using Cloud analyst simulator. Then we tested the proposed algorithm in a heterogeneous of processors power without considering network delay. Then we tested the proposed algorithm in heterogeneous of processors power with considering network delay. Finally we compared the results of the proposed algorithm with current algorithms results.

The algorithm adopts the characteristics of randomization and greedy to make an efficient load balancing and covers their disadvantages. The algorithm considers the current resource information and the CPU capacity factor to achieve the objectives. Figure 3.2 shows the abstract view of proposed a hybrid algorithm.

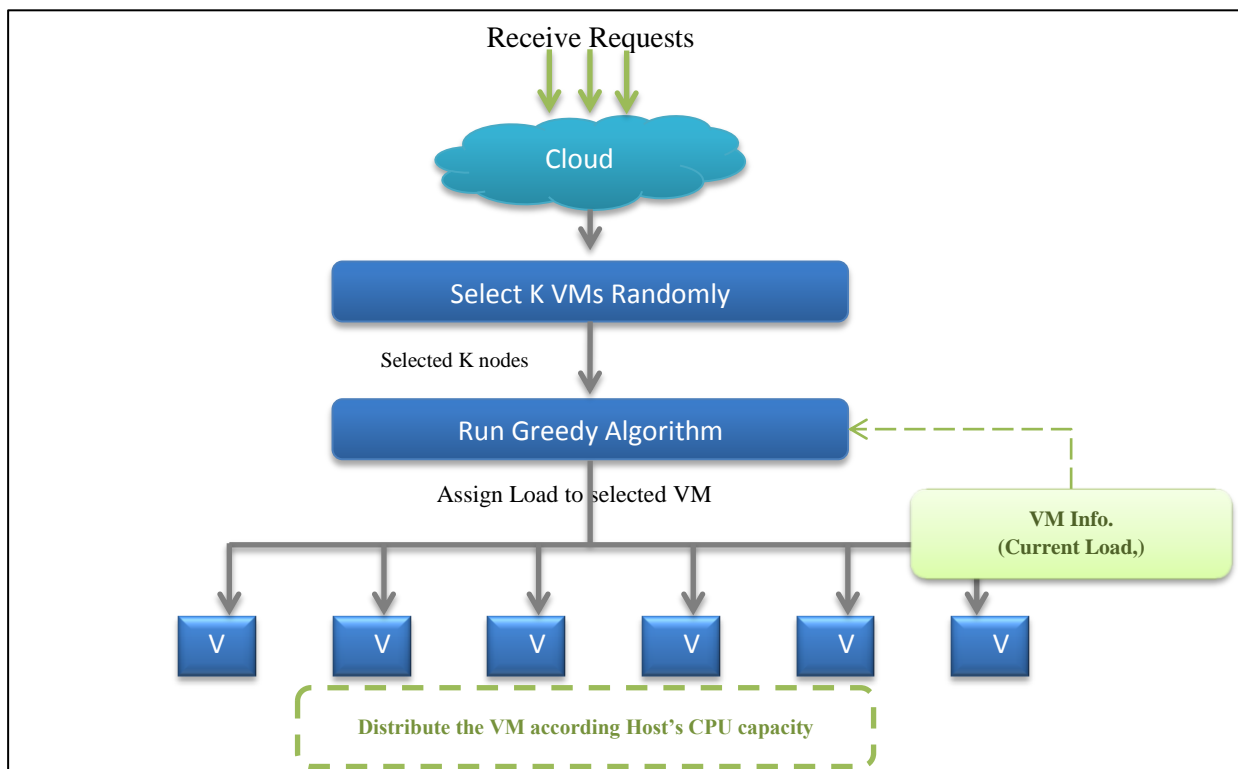


Figure 3.2 The Proposed Hybrid Algorithm

3.1.1 Description

The hybrid algorithm consists of two main steps which are:

- 1- In the first step VMs is distribute over hosts according to the host qualifications. The largest number of VMs is located at the most qualified host depending on the Hosts' CPU capacity. For example if we have five VMs and three hosts, where the first host has 1 CPU and its speed = 10000, the second host has 2 CPUs and the speed of every CPU = 10000, and the third host has 3 CPUs and the speed of every CPU = 100000. So, the capacity of the first host = $1*10000=10000$, the second host = $2*10000=20000$ and the third host = $3*10000=30000$. So according to hosts' capacities; first host will take 1 VM, the second host will take 2 VMs, and the third host which has the largest capacity will take 3 VMs.
- 2- In the second step the algorithm used a new index table to record the current loads for each VM. And which used to check the current loads for VM at each iteration, the algorithm read the value of VM load from the index table; when the data center receives a request from the users, it sends the request to the hybrid load balancer. The hybrid algorithm will select k nodes (VM) randomly, and then it will choose the current load for each selected VM. Then it will choose a VM that have least VM current loads and return the VM id to Data center. The Data center will assign the load to the selected VM and update the value of selected VM in the index table of current loads. Finally when the VM finishes processing the request, it will inform the data center to updating its current load value.

3.1.2 Implementation

The experimentation is done using the Cloud Analyst simulator [51]. We Define the simulator parameters such as (users Configuration, Data centers Configuration, VMs configuration), and we identified several Configurations. The experiments implemented using the identified configuration. In the first steps we studied the problem without the effect of network delay, we tested the algorithm in heterogeneous environment of hosts, and each machine has different number of CPUs and speed, and then we tested the effect of considering the Capacity of CPU factor. Finally we tested the impact of effect of network delay on the hybrid algorithm with considering the Capacity of CPU factor and in the heterogeneous environment of hosts. We implement some of current load balancing algorithms such as Round Robin, Equally spread current

execution, Random and Greedy algorithms. Then we implement the hybrid algorithm. The code of the hybrid algorithm is in appendix A.

3.1.3 Pseudo code

The Hybrid algorithm as given in Figure 3.3 is a load balancing algorithm used by the datacenter to distribute the received tasks efficiently over the virtual machine under a normal workload by finding the best VM among the group of VMs to assign the load in heterogeneous of a processors power in cloud computing environment. The hybrid algorithm consists of both random and greedy algorithms; the random algorithm which randomly select a VM to process the received tasks, does not need complex computation to make a decision but it does not select the best VM. On the other hand Greedy algorithm selects the best VM to handle the received task, but the selection process needs some complex computation to find the best VM. The hybrid algorithm considers the current resource information and the CPU capacity factor. The hybrid algorithm selects k nodes (VMs) randomly, and chooses the current load for each VM selected. Then the hybrid algorithm will choose a VM that have least VM current loads and return the VM ID to the Data center. The data center will assign the load to the selected VM and update the value of selected VM in the table of current loads. Finally when the VM finishes processing the request, it will inform the data center to update its current load value.

The Hybrid Algorithm

The Hybrid Load balancing algorithm uses randomization and greedy, it distributes the load over VMs to achieve efficient performance in heterogeneous cloud computing environment. The algorithm depends on current resource allocation count.

Input: list of VMs $VM_List()$, Maintain an index table of VMs with current allocation count for every VM $Cl_Table(VM_id)$, K where K is the number of VMs that will be selected randomly. $VMids()$ Maintain the index of selected node randomly with its current load, $TempVMid$ is a temp VMid that selected randomly

Output: VMid is the VM id that is selected to assign the load.

0. Distribute the VMs over the Hosts according to the host's qualification (VM provisioning).
1. Initialize, $Cl_Table(0..n-1) \leftarrow 0$ At start all VM's have zero allocation., $K \leftarrow m$, $VM_id \leftarrow -1$, $VMids() = -1, i \leftarrow 0$, $currCount \leftarrow 0$, $minCount \leftarrow Max_Value$, $TempVMid \leftarrow -1$;
2. Parses $VM_List()$ to LoadBalancer:
3. **For** $i \leftarrow 0$ to k //Select VM randomly
4. $TempVMid \leftarrow random(VM_List())$.
5. $VM_id \leftarrow TempVMid$
6. **If** vm_id Exist in $Cl_Table(VM_id)$ **then**
7. $currCount \leftarrow Cl_Table(VM_id)$
8. **Else**
9. $currCount \leftarrow 0$
10. $VMids() \leftarrow (VM_id, currCount)$.
11. **End for**
12. $TempVMid \leftarrow -1$
13. $currCount \leftarrow 0$
14. **For** $i \leftarrow 1$ to k
15. $TempVMid \leftarrow i$
16. $currCount \leftarrow VMids(TempVMid)$
17. **If** $currCount < minCount$ **then**
18. $minCount = currCount$
19. $VM_id \leftarrow TempVMid$
20. **End if**
21. **End for**

Figure 3.3 Hybrid Algorithm Pseudo Code

3.1.4 Evaluation

There are various metrics used to evaluate different techniques. In our work we used two metrics to measure the performance as follow:

- 1- **Response Time:** It is the time interval between sending a request and receiving its response. We should minimize the response time in order to enhance the system performance. The total response time can be obtained as follow:

$$\text{Total response time} = \text{the users request processing delay} + \text{Network delay} \dots (1)$$

- 2- **Processing time:** Average processing time: It is the amount of time actually needed to process a task.

3.2 Summary

In this chapter we presented a new hybrid algorithm based on randomization and greedy algorithm. The hybrid algorithm takes advantages of both random and greedy algorithms, and considers the current resource information and the CPU capacity factor to achieve the efficient performance in heterogeneous of a processors power in cloud computing environment.

We proposed a hybrid algorithm based on random and greedy algorithms. Then we implemented the proposed algorithm using Cloudanalyst simulator. After that we tested the proposed algorithm in heterogeneous of processors power without considering network delay. Then we tested the proposed algorithm in heterogeneous of processors power with considering network delay. Finally we compared the results of the proposed algorithm with current algorithms results.

Chapter 4

Experiments and results

Chapter 4 Experiments and Results

In this chapter we present the experiments and results that were done in this research. We have obtained the results by comparing our algorithm with some current load balancing algorithms. We used Cloud analysis simulator in the implementation.

As discussed in Figure 3.1. We studied the problem without the effect of network delay, then we tested the algorithm in a heterogeneous environment of hosts, and each machine has a different number of CPUs and speed. We tested the current algorithms' performance with light loads in a normal state, and then we tested the effect of considering the capacity of CPU factor. Finally we tested the impact of the effect of network delay on the hybrid algorithm with considering the capacity of CPU factor and in the heterogeneous environment of hosts; each machine has a different number of CPUs and speed. We compared the hybrid algorithm with Round robin, ESCE, Random and Greedy algorithms.

Section 1 will be about presenting the Experiments and results that were done in this research using cloudAnalyst simulator. Section 2 will be about presenting the summary of the experiments and results that were done.

4.1 Experiments

In the first step, we studied the problem without the effect of network delay, so all the user bases configuration will be in the same data center and in the same region.

4.1.1 Experiment 1: Test the hybrid algorithms without considering CPU capacity.

In this experiment we tested the algorithm in a heterogeneous environment of hosts; each machine has a different number of CPUs and speed.

4.1.1.1 Configuration

We defined the 50 virtual machines in the data center and the size used to host applications in the experiment is 100MB. Virtual machines have 1GB of RAM memory and have 10MB of available bandwidth. Simulated hosts have x86 architecture, virtual machine monitor Xen and Linux operating system. The data center hosts 5 virtual machines dedicated. The hosts have 2 GB of RAM and 100GB of storage. Each machine has a different number of CPUs and speed, first host has a 4 core processor with 2000 MIPS, second host has a 5 core 5000 MIPS, third host has a dual core with 9000 MIPS, fourth host has a dual core with 10000 MIPS, and fifth host has a dual core with 15000 MIPS.

Users are grouped by a factor of 1000, and requests are grouped by a factor of 100. Each user request requires 250 instructions to be executed. The simulation duration took one day. We used the response time and processing time metrics to compare the algorithm with other current algorithms. The configuration files as in Table 4.1, 4.2 and 4.3.

Table 4.1 Application development Configuration used in Experiment 1

Data Center	#VMs	Image Size	Memory	BW
DC1	50	10000	512	1000

Table 4.2 User bases configuration used in Experiment 1

Name	Region	Requests per user per Hr.	Data Size per req.(Bytes)	Peak Hours Start(GMT)	Peak Hours End(GMT)	Avg. Peak Users	Avg. Off-peak Users
UB1	0	12	100	13	15	400000	400000
UB2	0	12	100	15	17	100000	100000
UB3	0	12	100	20	22	300000	300000
UB4	0	12	100	1	3	150000	150000
UB5	0	12	100	21	23	500000	500000
UB6	0	12	100	9	11	800000	800000

Table 4.3 Data centers configuration used in Experiment 1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	2000	TIME_SHARED
1	204800	100000000	1000000	5	5000	TIME_SHARED
2	204800	100000000	1000000	2	9000	TIME_SHARED
3	204800	100000000	1000000	2	10000	TIME_SHARED
4	204800	100000000	1000000	2	15000	TIME_SHARED

4.1.1.2 Results

From this experiment we obtain results as in Table 4.4 and Figure 4.1:

Table 4.4 Response Time and processing time results without considering Capacity of CPU factor

Algorithms		Avg.(ms)	Min(ms)	Max(ms)
Round Robin	RT	871.76	60.89	3690.35
	PT	802.55	14.26	3524.49
ECSP	RT	872.64	60.89	4761.83
	PT	804.41	14.26	4696.53
Random	RT	887.85	51.61	6085.35
	PT	823.78	7.76	5982.06
Greedy	RT	873.60	60.89	3690.35
	PT	804.37	14.26	3524.49
Hybrid	RT	898.71	58.39	5824.38
	PT	830.19	12.57	5675.84

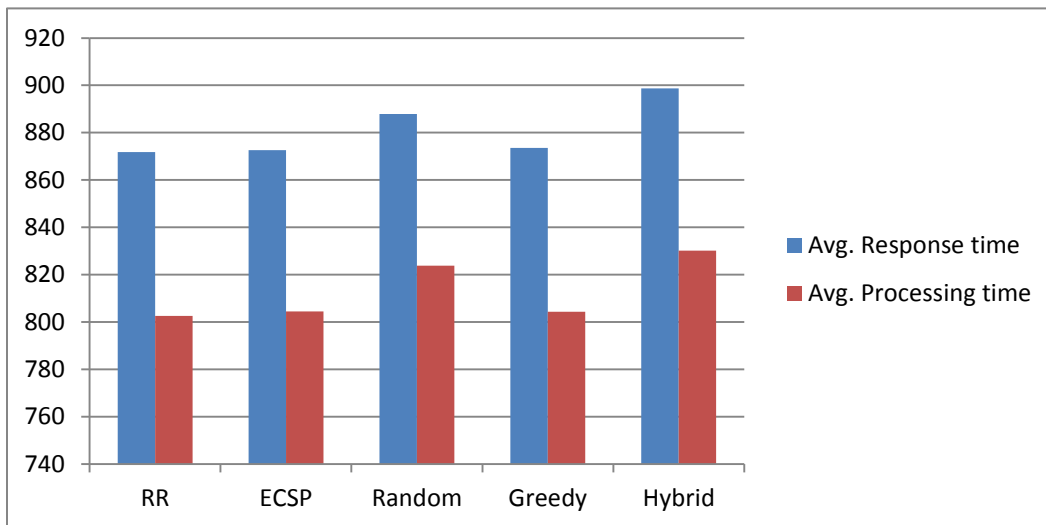


Figure 4.1 All algorithm results Comparison without considering Capacity of CPU factor

4.1.1.3 Discussion

The results showed that the Greedy, Round robin and ESCE had better results than the Hybrid algorithm and the random algorithm. This was due to the equivalent distributing of loads among all the VMs. Also we found that the Round Robin algorithm is better than the ESCE and Greedy algorithm because it is very simple and does not have the overhead computation as ESCE and Greedy.

In addition the random recorded the best min. response time 51.61 and the best min. processing time 7.76. Because the random does not need a complex computation to takes decision to allocate VM.

4.1.2 Experiment 2: Test the effect of considering the capacity of CPU.

This experiment studied the effect of considering the Capacity of CPU factor with the hybrid algorithm in heterogeneous environment; each machine has different number of CPUs and speed.

4.1.2.1 Configuration

The configuration files are as in experiment one in table 4.1, 4.2 and 4.3.

4.1.2.2 Results

From this experiment we obtain results as in Table 4.5 and in Figure 4.2:

Table 4.5 Response time and processing time results for testing the effect of Capacity of CPU factor

Algorithms		Avg. (ms)	Min(ms)	Max(ms)
Round Robin	RT	871.76	60.89	3690.35
	PT	802.55	14.26	3524.49
ECSP	RT	873.64	60.89	4761.83
	PT	804.41	14.26	4696.53
Random	RT	888.18	56.26	5819.01
	PT	824.10	7.60	5702.72
Greedy	RT	873.60	60.89	3690.35
	PT	804.37	14.26	3524.49
Hybrid	RT	747.24	48.76	8216.86
	PT	679.79	5.89	8084.95

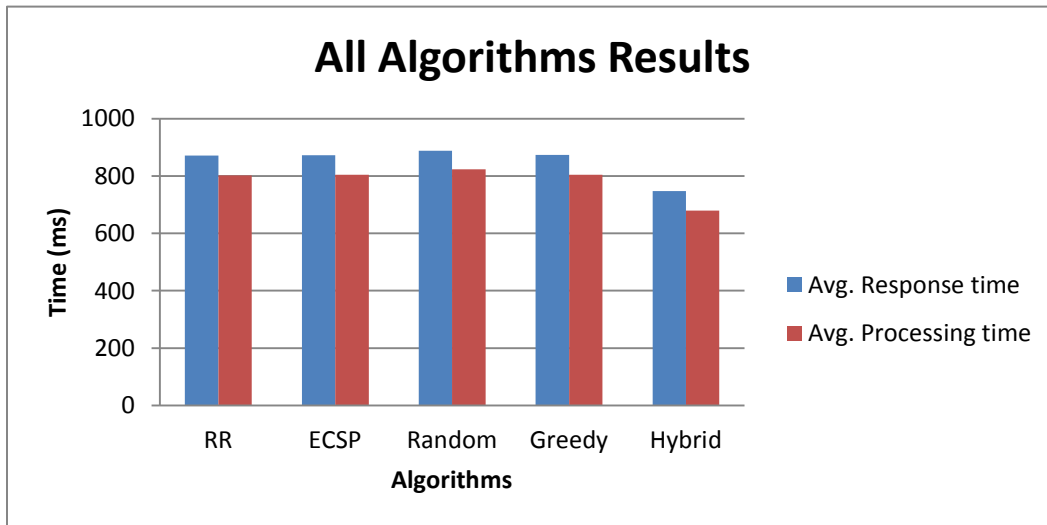


Figure 4.2 All algorithm results Comparison for testing the effect Capacity of CPU factor

4.1.2.3 Discussion

In this experiment the VMs distributed on the hosts according the hosts qualification and according the CPU capacity, the results showed that when considering the CPU capacity factor, the host that has best qualification has more VMs than other hosts, so when we select K nodes randomly from the VMs and choose the least loaded one from the selected VMs, the response time will be improved because most of VMs selected will be in the qualified host as seen in figure 4.3.

The hybrid algorithm recorded the best average response time **747.24** (ms) and the best average processing time **679.79** (ms) when K= 15. This result is better than round robin which had been the best algorithm results before. Round robin recorded average response time **871.76** (ms) and the average processing time **802.55** (ms). The difference between the results and other algorithms results exceeded **100** (ms) on each average response and processing time. This means minimizing the number of VM to 15 and with considering the CPU capacity factor, the hybrid algorithm decreased the overhead computation. The hybrid algorithm adds a significant improvement on average response time and on processing time compared with other algorithms. So, the hybrid algorithm improved the cloud computing performance in heterogeneous environment.

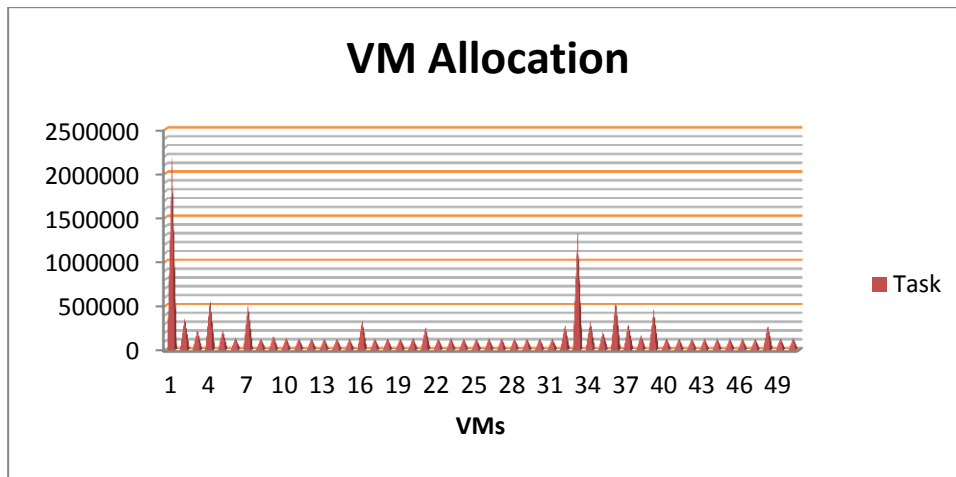


Figure 4.3 VM Allocations in Data center

When we look at figure 4.3 we can observe that the VMs which were in the high qualified host takes more tasks than the machine hosted in other hosts, for example VM 0 have 2176049 tasks and VM 34 have 1331363 tasks. The decision to allocate the tasks on the hosts improved and balanced.

In the second step, we will study the problem with considering the network delay by distributing user bases all over the world.

4.1.3 Experiment 3: tested the effect of network delay with considering the Capacity of CPU

This experiment tested the effect of network delay on the hybrid algorithm with considering the Capacity of CPU factor and in the heterogeneous environment of hosts. Each machine has different number of CPUs and speed.

4.1.3.1 Configuration

The configuration files are as in experiment one in Table 4.1, 4.2 and 4.3. And in order to test the effect of network delay we distributed the user's base in 6 regions, UB1 N-America, UB2 in S. America, UB3 Europe, and UB4 in Asia, UB5 Africa and UB6 in Oceania as in Table 4.6.

Table 4.6 User Bases Configuration Used in Experiment3

Name	Region	Requests per user per Hr.	Data Size per req.(Bytes)	Peak Hours Start(GMT)	Peak Hours End(GMT)	Avg. Peak Users	Avg. Off-peak Users
UB1	N-America	12	100	13	15	400000	400000
UB2	S. America	12	100	15	17	100000	100000

UB3	Europe	12	100	20	22	300000	300000
UB4	Asia	12	100	1	3	150000	150000
UB5	Africa	12	100	21	23	500000	500000
UB6	Oceania	12	100	9	11	800000	800000

4.1.3.2 Results

From this experiment we obtain results as in Table 4.7 and Figure 4.4:

Table 4.7 Response timer and processing time results for testing the effect of network delay

Algorithms		Avg.(ms)	Min(ms)	Max(ms)
Round Robin	RT	1050.20	74.78	3999.53
	PT	735.88	7.49	3415.15
ECSP	RT	1068.58	74.78	4999.80
	PT	754.69	7.50	4659.96
Random	RT	1067.78	71.12	6234.36
	PT	761.99	7.49	5681.28
Greedy	RT	1068.56	74.78	4963.67
	PT	754.68	7.50	4621.32
Hybrid	RT	930.77	62.79	8520.93
	PT	620.07	5.83	7983.89

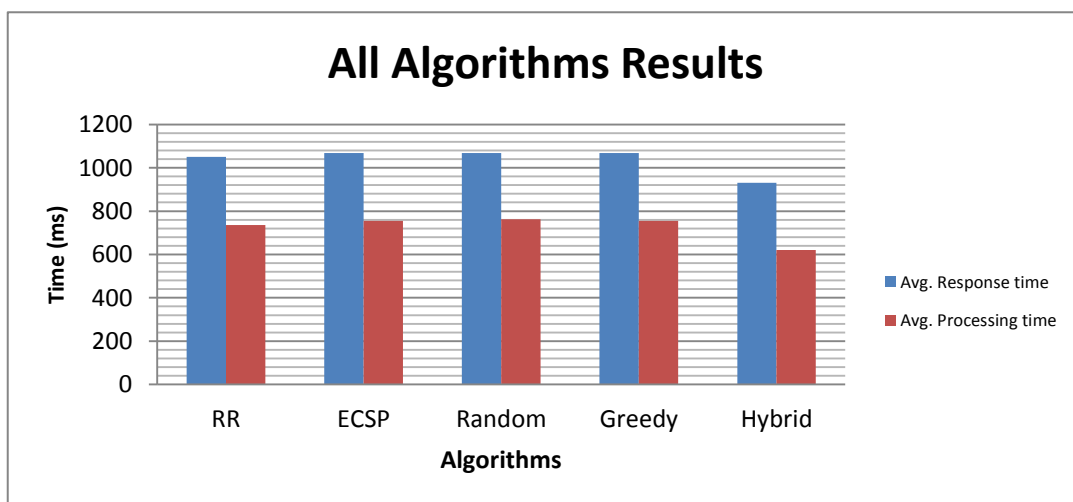


Figure 4.4 All algorithm results Comparison for testing the effect of network delay

4.1.3.3 Discussion

In this experiment we distributed the user in different areas to study the effect of network delay on the hybrid algorithm with considering the Capacity of CPU factor and in the heterogeneous environment of hosts. The results found that the network delay did not affect on the performance of hybrid algorithm, and the hybrid algorithm still recorded the best response time and processing time compared with other algorithms. The average response time was **930.77**(ms) and the average processing time was **620.07**(ms) when K= 20. This result was better than round robin which gave been the best algorithm results before. Round robin recorded average response time **1050.20**(ms) and the average processing time **735.88**(ms). The difference between the results and other algorithms results exceeded **100** (ms) on each average response and processing time. This means the hybrid algorithm add a significant improvement on average response time and on processing time with a network delay compared with other algorithms. The performance is improved in heterogeneous of a processors power in cloud computing environment.

4.2 Summary

In the first experiment we studied the problem without considering the network delay and without considering the CPU Capacity factor, the results shown that the Greedy, Round robin and ESCE had a better result than the Hybrid algorithm and the random algorithm. This was due to the equivalent distributing of loads between all the VMs.

On other hand in the second experiment when the VMs distributed on the hosts according to hosts qualification and CPU Capacity, the host that have best qualification had more VMs than other hosts, so when we select K nodes randomly from the VMs and choose the least loaded one from the selected VMs, the response time was improved because most of VMs selected have been in the qualified host. The hybrid algorithm without considering the network delay recorded the best average response time **747.24** (ms) and the best average processing time **679.79** (ms) when K= 15.

Finally, in the third experiment we studied the problem with considering the network delay. The results found that the network delay did not affect the performance of hybrid algorithm, and the hybrid algorithm still recorded the best response time and processing time compared with other algorithms. The average response time was **930.77** (ms) and the average processing time was

620.07 (ms) when $K=20$. The difference between others algorithms results exceeded 100 (ms) on each average response and processing time. This means the hybrid algorithm add a significant improvement on average response time and on processing time with a network delay compared with other algorithms. The performance has improved in heterogeneous of a processors power in cloud computing environment in normal state.

Chapter 5

Conclusion and Future Works

Chapter 5 Conclusion and Future Work

5.1 Conclusion

Load balancing is one of the important issues in cloud computing. The current load balancing scheduling algorithms in cloud computing environment have some deficiency and this would affect the performance. Therefore we proposed a hybrid algorithm to enhance the cloud computing performance. The hybrid algorithm based on randomization and greedy algorithm, they take the advantages of both random and greedy algorithms and consider the current resource information and the CPU capacity factor to achieve the objectives. The experiments were implemented in the CloudAnalyst Simulator.

Without considering the CPU Capacity factor, the results have shown that the Greedy, Round robin and ESCE had a better result than the Hybrid algorithm and the random algorithm. This was due to the equivalent distributing of loads between all the VMs. Also we found that the Round Robin algorithm was better than the ESCE and Greedy algorithm because it is simple and does not have the overhead computation as ESCE and Greedy

On the other hand when the VMs distributed on the hosts according the hosts qualification and according the CPU Capacity, the host that have best qualification will have more VMs than other hosts, so when we select K nodes randomly from the VMs and choose the least loaded one from the selected VMs, the response time will be improved because most of VMs selected will be in the qualified host.

The hybrid algorithm without considering the network delay recorded the best average response time **747.24** (ms) and the best average processing time **679.79** (ms) when K= 15. In addition the hybrid algorithm with considering the network delay recorded also a best response time **930.77** (ms) and a best processing time **620.07** (ms) when K= 20. This result was better than round robin which had been the best algorithm results before. Round robin without considering the network delay recorded average response time **871.76** (ms) and the average processing time **802.55**(ms) and with considering the network delay recorded average response time **1050.20** (ms) and the average processing time **735.88**(ms). The difference between the results exceeded **100** (ms) on each average response and processing time around **9.02 %**; this means the hybrid algorithm adds significant improvements on average response time and on processing time compared to other

algorithms. The performance has improved in heterogeneous of a processors power in cloud computing environment.

5.2 Future Work

Load balancing considered as one of the most challenges in cloud computing, it is the major factor to improve the performance of the cloud computing. We discussed only on improving the performance on one data center, but there are still other approaches that can be applied to balance the load in clouds computing environment with distributed Data centers. So we are going to implement a new load balance algorithm to improve the service broken policy. We tested only the effect of considering CPU capacity but there are other factors such as memory, bandwidth and storage. And we can also consider other parameters for efficient utilization of resources such as consider cost, failover etc. We studied the load balance in a normal state but there are still other state can be studied such as bursty load state. We are going to study on how we can overcome the problem of deadlocks and server overflow.

References

1. Florence, A.P. and Shanthi, V., *Intelligent Dynamic Load Balancing Approach for Computational Cloud*. International Journal of Computer Applications: pp. 15-18,(2013).
2. *Heterogeneous computing*. [cited 2015; Available from: http://en.wikipedia.org/wiki/Heterogeneous_computing.
3. Sharma, T. and Banga, V.K., *Efficient and Enhanced Algorithm in Cloud Computing*. International Journal of Soft Computing and Engineering (IJSCE), **3**(1),(March 2013).
4. Zhang, Q., Cheng, L., and Boutaba, R., *Cloud computing: state-of-the-art and research challenges*. Journal of Internet Services and Applications, **1**(1): pp. 7-18,(2010).
5. Singh, A., Gupta, S., and Bedi, R., *Comparative Analysis of Proposed Algorithm With Existing Load Balancing Scheduling Algorithms In Cloud Computing*. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), **3**(1): pp. 197-200,(2014).
6. Tiwari, M., Gautam, K., and Katara, K., *Analysis of Public Cloud Load Balancing using Partitioning Method and Game Theory*. International Journal of Advanced Research in Computer Science and Software Engineering, **4**(2): pp. 807-812,(2014).
7. Ratan, M. and Anant, J., *Ant colony Optimization: A Solution of Load Balancing in Cloud*. International Journal of Web & Semantic Technology (IJWesT), **III**,(2012).
8. Khatib, V. and Khatibi, E. *Issues on Cloud Computing : A Systematic Review*. in *International Conference on Computational Techniques and Mobile Computing*. Singapore, (2012).
9. Kulkarni, G., Gambhir, J., and Palwe, R., *Cloud Computing-Software as Service*. International Journal of Computer Trends and Technology, **2**(2): pp. 178-182,(2011).
10. Buyya, R., Ranjan, R., and Calheiros, R.N. *Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities*. in *International Conference on High Performance Computing and Simulation*. Proceedings of the 2009 International Conference on High Performance Computing and Simulation, HPCS 2009, (2009).
11. Kaleeswari and Juliet, N., *Dynamic Resource Allocation by Using Elastic Compute Cloud Service*. International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), **3**(5): pp. 12375-12379,(2014).
12. Jararweh, Y., Alshara, Z., Jarrah, M., Kharbutli, M., and Alsaleh, M.N. *TeachCloud: A Cloud Computing Educational Toolkit*. in *The 1st International IBM Cloud Academy Conference*. North Carolina, USA, pp. 1-16, (2012).
13. Sareen, P., *Cloud Computing: Types, Architecture, Applications, Concerns, Virtualization and Role of IT Governance in Cloud*. International Journal of Advanced Research in Computer Science and Software Engineering, **3**(3): pp. 533-538,(2013).
14. Shah, J., *Cloud Computing: The Technology for Next Generation*. International Journal of Advances in Computer Science and Technology, **3**(3): pp. 152-155,(2014).
15. Sajid, M. and Raza, Z. *Cloud Computing: Issues & Challenges*. in *International Conference on Cloud*. pp. 35-41, (2013).
16. Marisol, G.-V., Cucinotta, T., and Lu, C., *Challenges in real-time virtualization and predictable cloud computing*. Journal of Systems Architecture (ELSEVIER): pp. 1-15,(2014).
17. Kumar, S. and Aramudhan, M., *Performance Analysis of Cloud under different Virtual Machine Capacity*. International Journal of Computer Applications, **68**(8): pp. 1-4,(2013).
18. Ray, S. and De Sarkar, A., *Execution Analysis Of Load Balancing Algorithms In Cloud Computing Environment*. International Journal on Cloud Computing: Services and Architecture (IJCCSA), **2**(5): pp. 1-13,(2012).
19. Johnston, S. *Cloud Computing Types: Public Cloud, Hybrid Cloud, Private Cloud*. samj 2009 March 6; Available from: <http://samj.net/2009/03/cloud-computing-types-public-cloud.html>.
20. Vouk, M.A., *Cloud Computing – Issues, Research and Implementations*. Journal of Computing and Information Technology - CIT: pp. 235-246,(2008).
21. Padhy, R.P. and Rao, G.P., *Load Balancing in Cloud Computing Systems*: Orissa, India, (2011).

22. O., K.S., F., I., and O., A., *Cloud Computing Security Issues and Challenges*. International Journal of Computer Networks (IJCN), **3**(5): pp. 247-255,(2011).
23. Khanghahi, N. and Ravanmehr, R., *Cloud Computing Performance Evaluation: Issues And Challenges*. International Journal on Cloud Computing: Services and Architecture (IJCCSA), **3**(5): pp. 29-41,(2013).
24. Deepika, Wadhwa, D., and Kumar, N., *Performance Analysis of Load Balancing Algorithms in Distributed System*. Advance in Electronic and Electric Engineering, **4**(1): pp. 59-66,(2014).
25. Mohapatra, S., Rekha, K.S., and Mohanty, S., *A Comparison of Four Popular Heuristics for Load Balancing of Virtual Machines in Cloud Computing*. International Journal of Computer Applications, **68**,(2013).
26. Yao, J.H., Ju-hou. *Load Balancing Strategy of Cloud Computing Based on Artificial Bee Algorithm in Computing Technology and Information Management (ICCM)*. Seoul: IEEE, pp. 185 - 189, (2012).
27. Shameem, P.M. and Shaji, R.S., *A Methodological Survey on Load Balancing Techniques in Cloud Computing*. International Journal of Engineering and Technology (IJET), **4**(5): pp. 3801-3812,(2013).
28. Behal, V. and Kumar, A. *Cloud Computing: Performance Analysis of Load Balancing Algorithms In Cloud Heterogeneous Environment*. in *Confluence The Next Generation Information Technology Summit (Confluence)*. Noida: IEEE, pp. 200 - 205, (2014).
29. Kaushik, V.K., Sharma, H.K., and Gopalani, D. *Load Balancing in Cloud Computing Using High Level Fragmentation of Dataset*. in *International Conference on Cloud, Big Data and Trust*. pp. 118-126, (2013).
30. Sethi, S., Anupama, S., and Jena, K., S, *Efficient load Balancing in Cloud Computing using Fuzzy Logic*. IOSR Journal of Engineering (IOSRJEN), **2**(7): pp. PP 65-71,(2012).
31. Pathak, K.K., Yadav, P.S., Tiwari, R., and Gupta, T., *A Modified Approach for Load Balancing in Cloud Computing Using Extended Honey Bee Algorithm*. IJRREST: International Journal of Research Review in Engineering Science and Technology, **1**(3): pp. 12-19,(2012).
32. Mehta, R., Yask, P., and Harshal, T., *Architecture for Distributing Load Dynamically in Cloud Using Server Performance Analysis Under Bursty Workloads*. **1**(9),(2012).
33. Dave, S. and Maheta, P., *Utilizing Round Robin Concept for Load Balancing Algorithm at Virtual Machine Level in Cloud Environment*. International Journal of Computer Applications, **49**(4),(2014).
34. Mohialdeen, I.A., *Comparative Study of Scheduling Algorithms In Cloud Computing Environment*. Journal of Computer Science, **2**(9): pp. 252-263,(2013).
35. Cormen, T., Leiserson, C., Rivest, R., and Stein, C., *Introduction to Algorithms*. (2009).
36. Calheiros, R.N., Ranjan, R., De Rose, C.A.F., and Buyya, R., *CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*. pp. 1-9, (2009).
37. Pakize, S.R., Khademi, S.M., and Gandomi, A., *Comparison of CloudSim, CloudAnalyst And CloudReports Simulator in Cloud Computing*. International Journal of Computer Science And Network Solutions, **2**: pp. 19-27,(2014).
38. Wickremasinghe, B., Calheiros, R.N., and Buyya, R. *CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications*. in *24th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE Computer Society, pp. 446-452, (2010).
39. Mishra, R.K. and Bhukya, S.N., *Service Broker Algorithm for Cloud-Analyst*. International Journal of Computer Science and Information Technologies, **5** (3): pp. 3957-3962,(2014).
40. Limbani, D. and Oza, B., *A Proposed Service Broker Strategy in CloudAnalyst for Cost-Effective Data Center Selection*. International Journal of Engineering Research and Applications, **2**(1): pp. 793-797,(2012).
41. Ahmed, T. and Singh, Y., *Analytic Study of Load Balancing Techniques Using Tool Cloud Analyst*. International Journal of Engineering Research and Applications (IJERA), **2**(2): pp. 1027-1030,(2012).
42. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., and Buyya, R., *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*. Software: Practice and Experience, **41**(1): pp. 23-50,(2011).

43. Wickremasinghe, B., *CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments*, (2009).
44. James, J. and Verma, B., *Efficient Vm Load Balancing Algorithm for a Cloud Computing Environment*. International Journal on Computer Science and Engineering (IJCSE), **4**(09): pp. 1658-1663,(2012).
45. Hu, J., Gu, J., Sun, G., and Zhao, T. *A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment*. in *3rd International Symposium on Parallel Architectures, Algorithms and Programming*. IEEE, pp. 89-96, (2010).
46. Fang, Y., Wang, F., and Ge, J., *A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing*. Lecture Notes in Computer Science, **Jg. 2010**(6318): pp. 271-277,(2010).
47. Sharma, T. and Banga, V.K., *Proposed Efficient and Enhanced Algorithm in Cloud Computing*. International Journal of Engineering Research & Technology (IJERT), **2**(2),(2013).
48. Subramanian S, N.K.G., Kiran Kumar M, Sreesh P, and G R Karpagam, *An Adaptive Algorithm for Dynamic Priority Based Virtual Machine Scheduling in Cloud*. International Journal of Computer Science Issues, **9**(6),(2012).
49. Singh, A., Bedi, R., and Gupta, S., *Design and implementation of an Efficient Scheduling algorithm for load balancing in Cloud Computing*. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), **3**(1),(2014).
50. Singhal, U. and Jain, S., *A New Fuzzy Logic and GSO based Load balancing Mechanism for Public Cloud*. International Journal of Grid Distribution Computing, **7**(5): pp. 97-110,(2014).
51. *cloudsim*. cloudbus; Available from: <http://www.cloudbus.org/cloudsim/>.

Appendix A

1- Proposed Algorithm.

```
package cloudsim.ext.datacenter ;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import org.omg.CORBA.PRIVATE_MEMBER;
import cloudsim.VirtualMachine;
import cloudsim.VirtualMachineList;
import cloudsim.ext.Constants;
import cloudsim.ext.InternetCloudlet;
import cloudsim.ext.event.CloudSimEvent;
import cloudsim.ext.event.CloudSimEventListener;
import cloudsim.ext.event.CloudSimEvents;

public class VMLB extends VmLoadBalancer implements CloudSimEventListener
{
    public DatacenterController dcb;
    public InternetCloudlet cl;
    public Map<Integer, Integer> allocationCounts;

    public VMLB(DatacenterController dcb) {
        // TODO Auto-generated constructor stub
        dcb.addCloudSimEventListener(this);
        this.dcb=dcb;
        dcb.getVmCost();
        this.allocationCounts = getVmAllocationCounts();//number of allocations for
each VM.
    }
    @Override
    public int getNextAvailableVm() {
        // TODO Auto-generated method stub
        int temp=-1;
        int i;
        int k=30; // K is the number of VMs that will be selected randomly
        int vmId=-1;
        int currCount=0;
        int minCount = Integer.MAX_VALUE;
        Map<Integer, Integer> Mvmids = new HashMap<Integer, Integer>();//Maintain
the index of selected node randomly with its current load
        Random randomGenerator = new Random();
```

```

//Step 1: This command for Select (K) VMs randomly(random)

    for (i=0;i<k;i++)//Select 25 VM Randomly
    {
        //Generate random number between 0-50
        temp = randomGenerator.nextInt(dcbs.getVmStatesList().size());
        //check if this VM allocated before or not.
        if (currentAllocationCounts.containsKey(temp))
            currCount = currentAllocationCounts.get(temp);
        else
            currCount=0;
        //add the selected VMs_id and current allocating count. to a selected VMs table.
        Mvmids.put(temp, currCount);
    }
//end Step 1

//Step 2: Select VM with lest current Allocation Counts of Task (greedy)

    temp=-1;
    currCount=0;
    for (Iterator<Integer> itr = Mvmids.keySet().iterator(); itr.hasNext();)
        { //Select Vm_id from the table of a random Virtual Machines
            temp = itr.next();
        }
//Select the current count for each VM

        currCount = Mvmids.get(temp);
//check if this current count is the min. current count.

        if (currCount < minCount)
        {
            minCount = currCount;
            vmId = temp;
        }
    }

//End step 2

    allocatedVm(vmId);
    return vmId;
}
@Override
public void cloudSimEventFired(CloudSimEvent e)
{ // TODO Auto-generated method stub
    if (e.getId() == CloudSimEvents.EVENT_CLOUDLET_ALLOCATED_TO_VM)
    {
//this event indicate that the VM allocated and the current allocate count will increase by 1;
        int vmId = (Integer) e.getParameter(Constants.PARAM_VM_ID);
    }
}

```

```

        Integer currCount = currentAllocationCounts.remove(vmId);
        if (currCount == null)
        {
            currCount = 1;
        }
        else
        {
            currCount++;
        }
//Update the Value for allocated VM
        currentAllocationCounts.put(vmId, currCount);
    }
    else if (e.getId() == CloudSimEvents.EVENT_VM_FINISHED_CLOUDLET)
    {
//this event indicate that the VM deallocated and the current allocate count will decrease by
1;
        int vmId = (Integer) e.getParameter(Constants.PARAM_VM_ID);
        Integer currCount = currentAllocationCounts.remove(vmId);
        if (currCount != null)
        {
            currCount--;
//Update the Value for allocated VM
            currentAllocationCounts.put(vmId, currCount);
        }
    }
}

```