MULTI-OBJECTIVE TASK SCHEDULING USING K-MEANS ALGORITHM IN CLOUD COMPUTING

CHUE THEINGI KYAW

M.C.Sc. **JUNE 2022**

MULTI-OBJECTIVE TASK SCHEDULING USING K-MEANS ALGORITHM IN CLOUD COMPUTING

By

CHUE THEINGI KYAW

B.C.Sc.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Computer Science (M.C.Sc.)

University of Computer Studies, Yangon

JUNE 2022

ACKNOWLEDGEMENTS

To complete this thesis, many things are needed like my hand work as well as the supporting of many people.

Firstly, I would like to express my appreciation and sincere thanks to **Dr. Mie Mie Khin,** Rector of the University of Computer Studies, Yangon, for her kind permission to submit this dissertation.

I would like to thank course coordinators**, Dr. Si Si Mar Win** and **Dr. Tin Zar Thaw**, Professors, Faculty of Computer Science, the University of Computer Studies, Yangon for their superior suggestions and administrative supports during my academic study.

I wish to thank **Daw Mya Hnin Mon**, Associate Professor, English Department, the University of Computer Studies, Yangon, for editing my thesis writing from language point of view.

I am extremely grateful to my supervisor **Dr. Yu Mon Zaw**, Lecturer, Faculty of Computer Science, the University of Computer Studies, Yangon for their valuable guidance, supervision, patience, encouragement and editing during the period of study towards completion of this piece of research work.

I especially thank my parents, all of my colleagues, and friends for their encouragement and help during my thesis.

Finally, I would like to thank all of the staff, teachers from University of Computer Studies, Yangon for their support.

STATEMENT OF ORIGINALITY

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

---------- ----------

Date Chue Theingi Kyaw

ABSTRACT

The growth of information technology has led to the increasing need of computing and storage. Cloud services are the technologies with high demand, consisting of a set of virtualized resources that serve the users as per demand and on a pay-per-use basis via the internet. So, it must have the ability to meet all the users' requests with high performance and efficiency. Task scheduling is a very important aspect to improve the overall performance of the cloud computing. There are various types of scheduling algorithms for resource utilization. The poor task scheduling resources are not utilized efficiently. In order to address these weaknesses, this work proposes a novel approach to map groups of tasks to customized virtual machine types. Mapping of the tasks is based on task usage patterns like length, file size and bandwidth. The proposed system uses K-means clustering algorithm to generate the task clusters .The tasks are grouped base on their "Tasks Length and Deadline". After clustering, the individual task in each cluster is scheduled to appropriate VMs. The proposed system performs multi-objective task scheduling with an aim in minimizing the makespan and execution time. The experimental results show that the proposed method produces better results in terms of execution time, makespan than the traditional algorithm (FCFS). In this system, CloudSim simulator version 3.0.3, one of the open source frameworks is used .It is written in Java programming language.

Keywords: Cloud computing, Task Scheduling, Clustering, Execution time, Makespan

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1
INTRODUCTION

Nowadays, cloud computing has become a modern technology among the users, and the volume of data is increasing day by day. Cloud computing refers to the resources and services that are made available over the internet, with these services being incorporated into the cloud environment to meet user needs. The growths of cloud resources or services, as well as their increased use by people from varied backgrounds, necessitate a focus on cloud user management and their needs. The main objective of cloud computing is to combine multiple resources (hardware and software) that are available through the internet so that customers get all or some of these resources according to the used cloud system. Cloud computing has recently expanded significantly as a result of the employment of improved equipment, virtualization technologies, distributed systems, and other factors.

The rapid expansion of cloud computing services and the rising number of users necessitate the management of all cloud users and their requests. Task Scheduling is one of the most significant aspects of maintaining user requests in the cloud environment; it is used to improve the system performance and reduce the task performance time. A task scheduler technique is recognized as a basic requirement for providing a successful service in cloud computing. In the cloud datacenters, task scheduling is the process of mapping and assigning incoming tasks to available resources, where a resource is defined as a virtual machine (VM). The main focus of task scheduling is to enhance the efficient utilization of resources and reduce the task completion time. A good scheduling algorithm yields good system performance. Most task scheduling algorithms use single criteria that do not provide an efficient result for resource utilization. The proposed system uses two criteria to reduce the execution time and makespan. In this system, the multi-objective task scheduling technique is integrated with the K-means clustering algorithm for task allocation to virtual machines.

1.1 Objectives of the Thesis

The main goal of this system is to support the minimum execution time and makespan on cloud environments. The additional objectives are as follows:

- To apply the K-means clustering algorithm in cloud computing for task scheduling.
- To understand parallel computing which is the simultaneous use of multiple computing resources.
- To improve the resource utilization.

1.2 Related Works

In paper [1], the authors created the grouped tasks scheduling (GTS) method, which is used to schedule jobs in a cloud computing network utilizing Quality of Service (QoS) to meet the needs of users. This concept recommended grouping tasks into five categories, each of which contained tasks with similar qualities (task type, task size, task latency, and user type). The purpose of this paper was to obtain the shortest feasible execution time for all processes while maintaining minimal latency for high-priority processes.

In paper [2], the authors proposed a dynamic cloud task scheduling (DCTS) algorithm to classify the tasks and create the necessary VMs in advance, based on the scheduling history. In this way, it reduced the time to create the VMs and hence minimized the completion time of the tasks. The Bayes classifier was used to classify the tasks in this algorithm.

In paper [3], the creators proposed task scheduling algorithm is presented to schedule a task based on two parameters: task length and deadline. The purpose of this algorithm was to decrease the task completion time. When compared to traditional methods, the results reveal a reduction in the makespan and average waiting time.

In paper [4], the authors presented a multi-objective task scheduling algorithm for mapping tasks to VMs to improve the throughput of the datacenter and reduce the cost. The priority of a task is determined by the task's quality of service. The multi-objective task scheduling technique was implemented in this paper using non-dominated sorting.

In paper [5], the creators developed a selective algorithm that uses the Min-Min and Max-Min algorithms. It was determined to select one of these two algorithms, depending on the standard deviation of the expected completion times of the tasks on each of the resources. When compared to the FCFS algorithm, it was clear that this technique was more effective in minimizing the makespan.

In paper [6], the authors deployed cloud computing to implement the K-Means clustering algorithm. The major goal of this work was to use Google App Engine and Cloud SQL to implement and deploy the K-Means algorithm in the Google Cloud. The Google App Engine Plug-In is used to deploy the application on Google, and Google Cloud SQL is used to create the database and tables.

There is still a need for new solutions to further reduce the makespan and perform load balancing, despite the fact that the research community has made major contributions to solving the scheduling problem in cloud computing. Hence, a new scheduling technique namely multi-objective task scheduling using K-means algorithm is proposed to minimize the makespan and execution time.

1.3 Motivation of the Thesis

The rapid growth of cloud computing services and increasing numbers of users have become the dramatic need for modernized mechanisms for resource handling and task scheduling. Although many task scheduling algorithms have been developed by the research community towards providing a solution for the scheduling problem in cloud computing, there is still a need for new solutions to reduce the task completion time. Therefore, the proposed algorithm should have the ability to improve resource utilization and to support minimum execution time and makespan on cloud environments.

In the proposed system, K-means clustering algorithm is used for task scheduling in cloud computing. By applying K-means in task scheduling process, the proposed system performs multi-objective task scheduling with the aim of minimizing the makespan and execution time.

1.4 Overview of the Proposed System

In this thesis, multi-objective task scheduling using K-means algorithm is used to apply task scheduling in cloud computing. The K-Means method is used to group the tasks depending on their length and deadline. Then, resulting task clusters are mapped to the best VMs using Resource Allocation algorithm. Verification for the efficiency of the algorithm is done by testing it in terms of the QoS metrics such as makespan and execution time. The proposed algorithm is compared with the existing task scheduling algorithm (First Come First Serve) based on two criteria: makespan and execution time.

1.5 Organization of the Thesis

This thesis is organized as five chapters. They are as follows:

In Chapter 1, the general concept of cloud computing is described. It is followed by a discussion of the main objectives of this thesis, which include the scheduling tasks in cloud computing and presenting a suitable solution based on the proposed system. It also includes the examples of related work to this thesis. In Chapter 2, the concept of cloud computing is explained, which includes cloud task scheduling, cloud computing strategies, and cloud computing benefits. In Chapter 3, the proposed methodologies in details are discussed. In Chapter 4, the experimental setup, results, and comparisons between the existing algorithm (FCFS) and the proposed system are described in details. In Chapter 5, the outcomes of this system are concluded in this chapter and the future scopes related to this thesis are also offered.

CHAPTER 2
THEORETICAL BACKGROUND

Cloud computing is a rapidly expanding field. This technological advance has not only changed the way businesses work, but it has also dissolved the old IT department structure and placed new demands on employees who are redefining their careers. Jobs are rising at a fast speed as enterprises rapidly move their infrastructures and services to the cloud, with many roles staying unfilled [1].

2.1 Cloud Computing

A new business strategy for providing resources and services over the Internet is termed as cloud computing. Cloud computing is also a synthesis of numerous technologies, including virtualization, distributed application architecture, grid computing, utility computing, and clustering. Because of its economic benefits, cloud computing is the most widely utilized technique. In a cloud, all information technology services are given as a service to the user. Moreover, as the number of cloud users is increasing the consumption of energy by the cloud is also increasing. Microsoft Azure, Amazon EC2, Google App Engine, and Aneka have all recently become cloud-based environments [4].

On these virtual machines, cloud clients can run any software. This makes it easier for end users to use cloud services. With the progression of time, the scale of cloud computing has expanded, making it possible to access data from anywhere without being limited by a geographical location. The administration difficulty grows as the volume of data grows. Virtualization and the code layers, on the other hand, assist in this regard, providing for powerful server control. The system of cloud computing is shown in Figure 2.1.

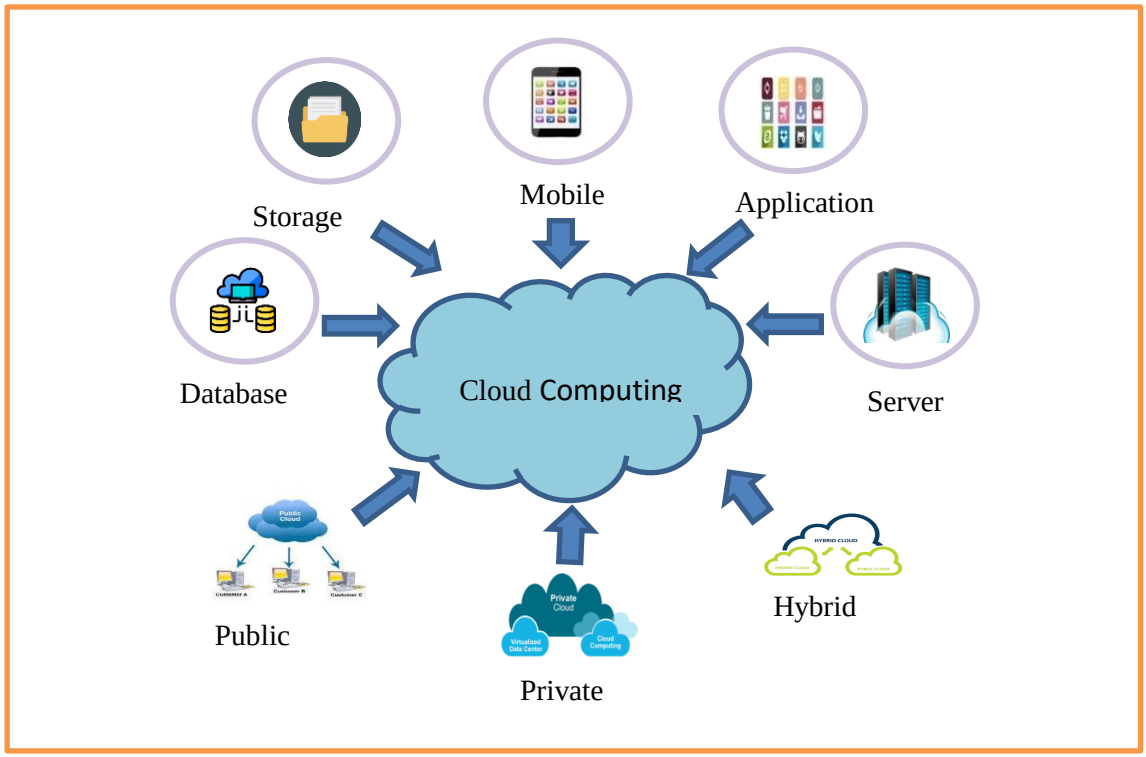

Figure-2.1 System of Cloud Computing

2.1.1 Types of Cloud Computing Services

Cloud is a huge data center with virtualization technology. All the infrastructures and services provided by the cloud can be accessed remotely. Depending on the services it provides, the cloud can be categorized into three types, Infrastructure as a Service [IaaS], Platform as a Service [PaaS] and Software as a Service [SaaS] [9]. Figure 2.2 gives the overview of cloud service models.

i. **Software as a Service:** Email is an example of software as a service. Cloud provides users the entire features to host an application. Security is much concern in SaaS and many providers take measures to protect the data of the application. Many mobile Apps use SaaS to host the application and use the storage of the Cloud [9].

ii. **Platform as a Service:** Entire platform is provided to the users, where they have control over the applications configuration, deployment and on maintenance. Cloud typically provides all the details like connectivity, scalability to the users. It acts as an application container to the user [9].

iii. **Infrastructure as a Service:** This is a service, which provides the basic blocks of infrastructure and resources such as network and storage. Cloud resources are provided virtually over the Internet. IaaS usually used for webhosting where the operating system is installed on the virtual machine. IaaS provide cloud features such as dynamic scaling, load balancing and global availability [9].

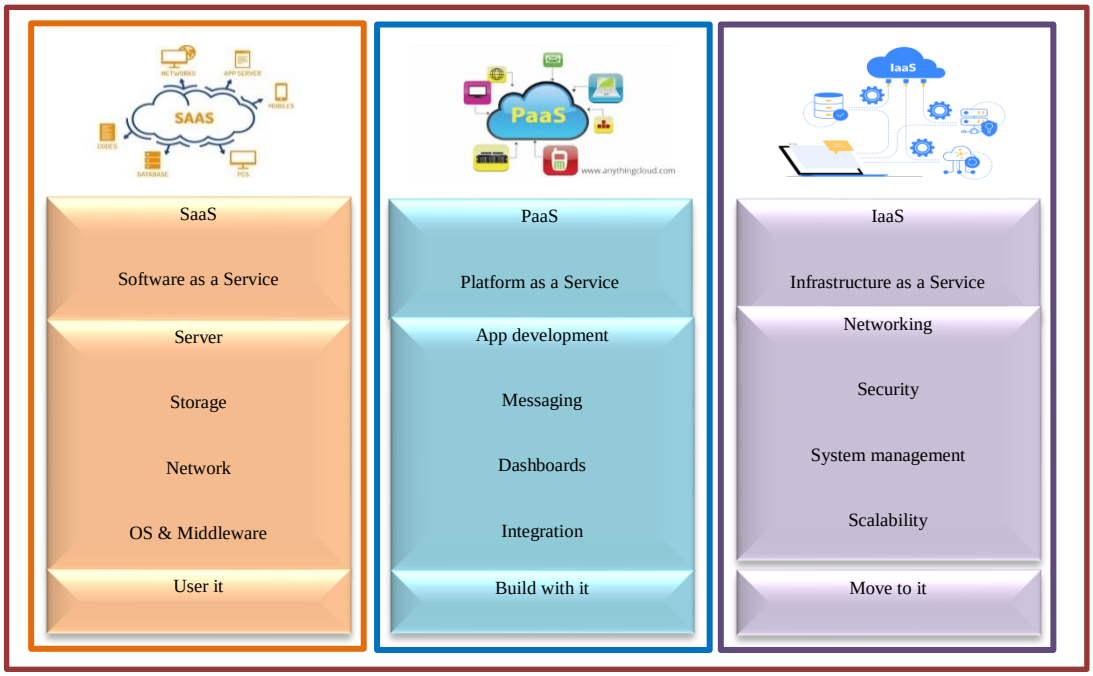

Figure-2.2 Cloud Service Models

2.1.2 Cloud Deployment Models

Cloud deployment models describe the nature of the cloud. Usually there are four types of deployment models, but due to the vast usage and requirement, additional two types of deployment models came into existence. Private Clouds: Here the cloud infrastructure is exclusively given to a single organization. It may be owned and maintained either by the organization or by any third party.

- Public Clouds: A cloud infrastructure in this type of cloud is open to public usage. It may be owned and maintained by either any business organization or by government organization.

- Hybrid Clouds: This type of cloud is the most economical cloud which is the combination of both public and private clouds. It combines the benefits of the controlled environment in private cloud and elasticity of public clouds.
- Community Clouds: Community cloud is slightly different from public cloud. This cloud provides exclusive access to the users of specific community from organizations that might share similar concerns and missions.
- Federated Clouds: This is one of the emerging cloud models. Like the internet, this is the mechanism which enables the users to utilize multiple cloud platforms from multiple cloud providers. Cloud Federation model allows sharing between 5 the clouds and combined provisioning.
- Inter-Clouds: Like Federated cloud model, Inter clouds are also emerging models. Inter cloud model provides a framework for multi-cloud provider for its services, infrastructure and operations. According to Figure 2.3, there are four deployment models available to end users.

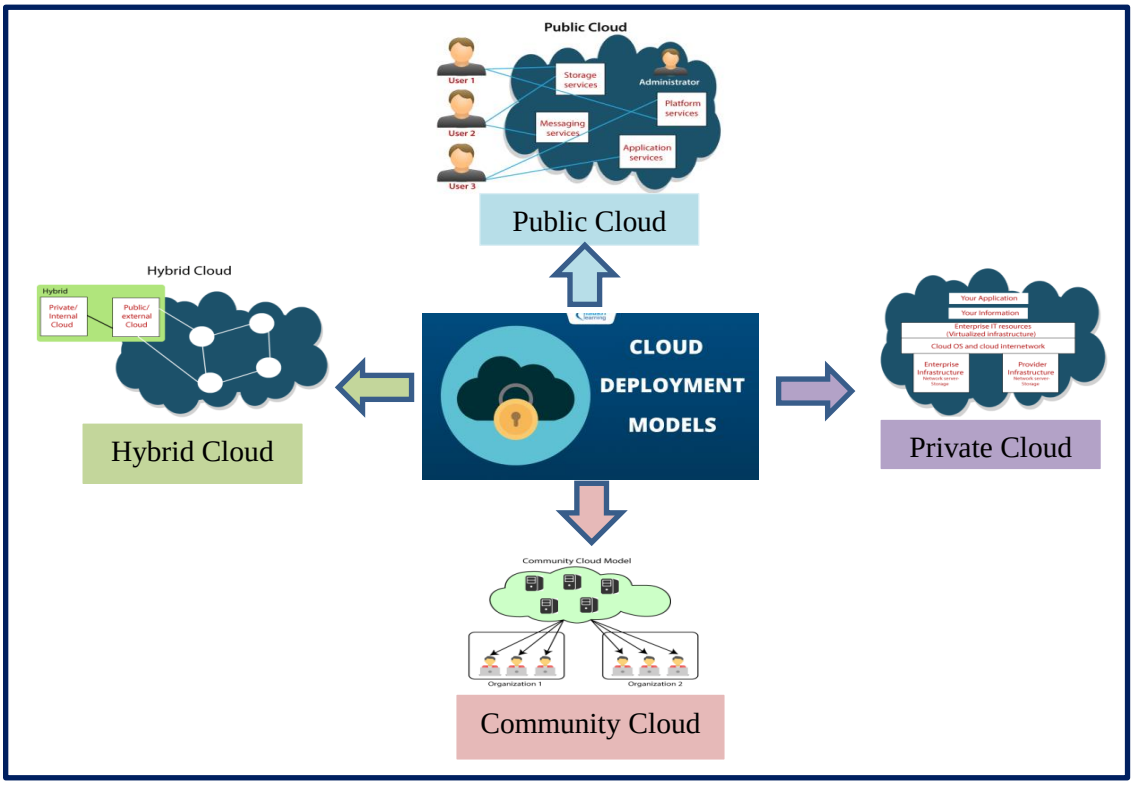

Figure-2.3 Cloud Deployment Models

2.1.3 Benefits of Cloud Computing

Cloud computing offers various advantages that help users adapt to the cloud. The most important advantage is that it reduces costs of IT infrastructure installation and management because cloud users only need a terminal to connect to the cloud and do not need to purchase any spatial hardware [22].

It is also distinguished by easier implementations, reduced initial and ongoing expenses. Pay only for the services you require, with a guaranteed service level agreement. Predictable spending, access to the cloud from anywhere, and decreased capital costs, Concentrate on business rather than technology, and so on [12]. Because data storage is not server-specific, the cloud is dependable for backup and recovery.

2.2 Task Scheduling

Scheduling is the process of determining the order in which resources are mapped to be executed. In cloud computing, task scheduling can define an order for tasks that achieves a balance between improving QoS and maintaining efficiency and fairness among tasks by selecting the best available resource for task execution or allocating virtual machines to tasks in such a way that the completion time is as minimal as possible [17] [18].

The following are the benefits of task scheduling algorithms: [10]

- Maintain cloud computing performance and service quality.
- Control CPU and memory usage.
- Effective scheduling techniques that optimize resource utilization.
- Improving overall task fairness.
- Increasing the amount of tasks completed successfully.
- Using a real-time system to schedule tasks.
- Improving load balancing.
- Increasing system throughput.

2.2.1 Task Scheduling Algorithms

The following are some of the most common scheduling algorithms: Priority Scheduling Algorithm, First Come First Serve Scheduling Algorithm, and Genetic Algorithm [2].

1. **First Come First Serve Scheduling Algorithm (FCFS):** This is a simple scheduling algorithm in which processes are ordered according to arrival time and submitted to the CPU.
2. **Priority Scheduling Algorithm:** This is a preemptive scheduling algorithm, which is based on priority.
    - **Shortest Job First Scheduling Algorithm (SJF) or (Min-Min algorithm):** It achieves the shortest average waiting time by scheduling a short task before a long one. It might be either preemptive or non-preemptive. A preemptive SJF method will terminate the current process, but a non-preemptive SJF approach will let the task finish its CPU burst.
    - **Round-Robin Scheduling Algorithm (RR):** Another method of priority scheduling. This is the simplest, most equitable, most extensively utilized scheduling algorithm. The smallest unit of time, known as time slices or quantum, is used to run all processes in a circular queue.
    - **Max-Min algorithm:** It is the opposite of SJF in that it prioritizes the larger tasks for completion first.
3. **Genetic Algorithm (GA):** It is a problem-solving strategy based on the genetics model. GA is a search technique for finding an optimal solution. [13].

2.2.2 Scheduling Process in Cloud Computing

In cloud computing, the major goals of task scheduling algorithms are to reduce turnaround time and maximize resource utilization. A major role was played in how to meet the needs of cloud users in terms of QoS and how to effectively employ cloud resources at a minimal cost. The main reason for scheduling is to improve the completed task at the lowest possible cost while maintaining the same level of QoS for the user.

The scheduling processes of the cloud are divided into three stages namely:

- **Resource discovery and filtering:** the resources provided in the network system and the status information collected about them by the Data center broker.
- **Resource selection:** the resource determined target depending on task and resource requirements.
- **Task allocation:** A task is assigned to choose the resource.

2.2.3 Classification of Task Scheduling Algorithms

In cloud computing, there are various task scheduling classifications. The following task scheduling algorithm types are depicted in Figure 2.4:

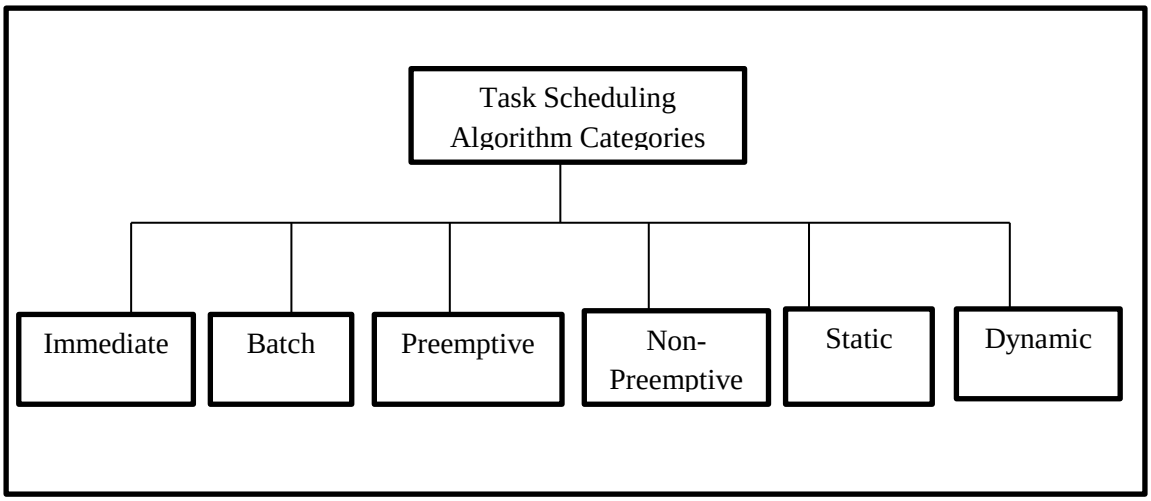

Figure-2.4 Task Scheduling classes

- Immediate scheduling: When new tasks arrive, they are assigned to VMs immediately.
- Batch scheduling: The tasks are grouped together before being dispatched in a batch; this is also known as mapping events.
- Static scheduling: It is regarded as simpler than dynamic scheduling even though it is based on prior knowledge of the system's global state. It discards VM state and then distributes all traffic equally among all VMs in a similar way to round robin (RR) and random scheduling approaches.

- Dynamic scheduling: It considers the present condition of VMs without requiring prior knowledge of the overall state of the system, and distributes workloads based on the capacity of all available VMs [1].
- Preemptive scheduling: Each task is paused during execution and can be shifted to another resource to finish [1].
- Non-preemptive scheduling: VMs are not reallocated to new tasks until the specified task is completed [1].

2.3 First Come First Serve (FCFS) Algorithm

In this approach, the order of tasks in the task list is determined by their arrival time and then assigned to appropriate VMs. [2].

2.3.1 Advantages and Disadvantages of FCFS

Advantages

- It is the most widely used and simplest scheduling algorithm.
- It is more equitable than other scheduling algorithms.
- It uses the FIFO rule when scheduling tasks.
- It is simpler than other scheduling algorithms.

Disadvantages

- It does not give any priority to tasks.
- The tasks have a high waiting time. This means that if the beginning tasks list contains large tasks, all tasks must wait a long time for the large tasks to be completed.
- The resources are not being consumed optimally.

2.3.2 Example Calculations of FCFS

When assigning the tasks to virtual machines (VMs) in a cloud computing environment, some assumptions must be considered.

- Number of tasks should be more than the number of virtual machines (VMs), implying that each VM must perform several tasks.
- Each task is allocated to a specific virtual machine resource.
- Virtual machines are resource and control independent.
- Task lengths are arranged according to small, medium, and large.
- Execution of a task is not interrupted once it begins.

The available VMs are for exclusive use only and cannot be shared between tasks. It indicates that the VMs cannot consider about anything else until the current tasks are completed [7].

In Table 2.1, the fifteen tasks with different lengths are shown in the following:

Table 2.1 Set of tasks with different lengths

Task	Length
T1	100000
T2	70000
T3	5000
T4	1000
T5	3000
T6	10000
T7	90000
T8	100000
T9	15000
T10	1000
T11	2000
T12	4000
T13	20000
T14	25000
T15	80000

As an example, six VMs with different properties based on tasks size: VM lists = {VM1, VM2, VM3, VM4, VM5, VM6}. Million instructions per second (MIPS) of VM lists = {500, 500, 1500, 1500, 2500, 2500}.

Figure 2.5 depicts the working mechanism of the FCFS tasks scheduling algorithm and how tasks are completed based on their arrival time.

- The initial set of tasks is scheduled based on their arrival time, as illustrated by the blue arrows.
- The second set of tasks is scheduled based on their arrival time, as indicated by the red arrows.
- The third set of tasks is scheduled based on their arrival time, as represented by the black arrows.

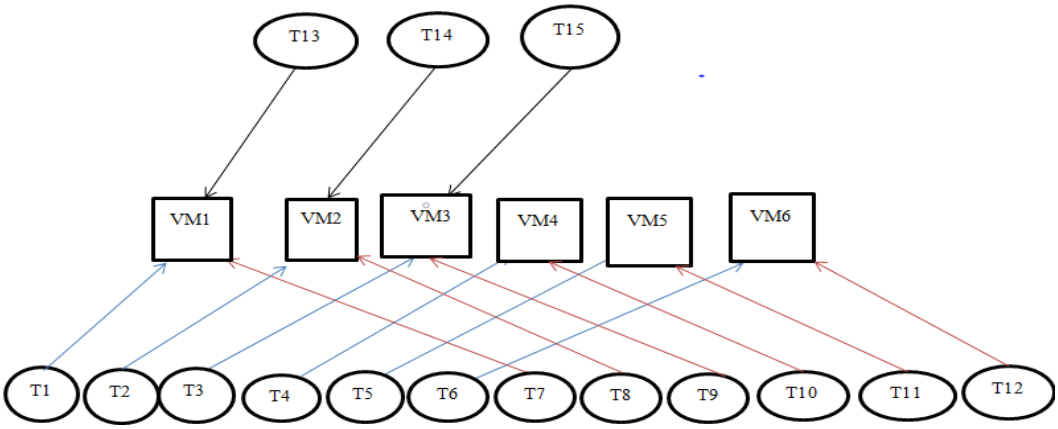

Figure-2.5 FCFS work mechanism

In this case, T1 is clearly excessively large in comparison to T7 and T12. However, as T7 and T12 must wait for T1, the Task Waiting Time, Task Execution Time, and Task Finish Time are all increased, and reduce fairness.

The tasks assigned to the VMs list sequentially are shown in the following:

$$VM1 = \{T1 \longrightarrow T7 \longrightarrow T12\}$$
$$VM2 = \{T2 \longrightarrow T8 \longrightarrow T14\}$$
$$VM3 = \{T3 \longrightarrow T9 \longrightarrow T15\}$$
$$VM4 = \{T4 \longrightarrow T10\}$$
$$VM5 = \{T5 \longrightarrow T11\}$$
$$VM6 = \{T6 \longrightarrow T13\}$$

The FCFS scheduling technique increases waiting time for all tasks can be seen in Table 2.2.

Table 2.2 Task Waiting Times in FCFS

Task	Length	Execution Time	Waiting Time	
T1	100000	200	VM1	
T2	70000	140	VM2	
T3	5000	3.33	VM3	
T4	1000	0.66	VM4	
T5	3000	1.2	VM5	
T6	10000	4	VM6	
T7	90000	180	Wait(200)	VM1
T8	100000	200	Wait(140)	VM2
T9	15000	10	Wait(3.33)	VM3
T10	1000	0.66	Wait(0.66)	VM4
T11	2000	0.8	Wait(1.2)	VM5
T12	4000	1.6	Wait(4)	VM6
T13	20000	40	Wait(200+180=380)	VM1, VM7
T14	25000	50	Wait(140+200=340)	VM2,VM8
T15	80000	53.33	Wait(3.33+10=13.33)	VM3,VM9

2.4 Shortest Job First (SJF) Algorithm

The order of tasks is determined by their priority. Priority is assigned to tasks based on their length, with the smallest task receiving the highest priority.

2.4.1 Advantages and Disadvantages of Shortest Job First (SJF)

Advantages

- SJF has minimum average waiting time among all tasks scheduling algorithms.
- Waiting time is lower than FCFS.

Disadvantages

- When tasks are assigned to VM, some tasks are unfairly treated since large tasks are left waiting in the task list while small tasks are assigned to VM.
- It takes a long time to execute and complete the task.

2.4.2 Example Calculations of SJF

The work mechanism for SJF will be as follows:

Table 2.3 shows how the tasks described in Table 2.1 are categorized from smallest to largest depending on their lengths

Table 2.3 A set of tasks sorted based on SJF scheduling algorithm

Task	Length
T4	1000
T10	1000
T11	2000
T5	3000
T12	4000
T3	5000
T6	10000
T9	15000
T13	20000
T14	25000
T2	70000
T15	80000
T7	90000
T1	100000
T8	100000

The tasks allocated to the VMs list are defined in the following order:

$$VM1 = \{T4 \longrightarrow T6 \longrightarrow T7\}$$
$$VM2 = \{T10 \longrightarrow T9 \longrightarrow T1\}$$
$$VM3 = \{T11 \longrightarrow T13 \longrightarrow T8\}$$
$$VM4 = \{T5 \longrightarrow T14\}$$
$$VM5 = \{T12 \longrightarrow T2\}$$
$$VM6 = \{T3 \longrightarrow T15\}$$

According to Table 2.4, the larger tasks must wait in the task list until the smaller tasks complete execution.

Table 2.4 Task Waiting Times in SJF

Task	Length	Execution Time	Waiting Time	
T1	100000	200	VM1	
T2	70000	140	VM2	
T3	5000	3.33	VM3	
T4	1000	0.66	VM4	
T5	3000	1.2	VM5	
T6	10000	4	VM6	
T7	90000	180	Wait(200)	VM1
T8	100000	200	Wait(140)	VM2
T9	15000	10	Wait(3.33)	VM3
T10	1000	0.66	Wait(0.66)	VM4
T11	2000	0.8	Wait(1.2)	VM5
T12	4000	1.6	Wait(4)	VM6
T13	20000	40	Wait(200+180=380)	VM1,VM7
T14	25000	50	Wait(140+200=340)	VM2,VM8
T15	80000	53.33	Wait(3.33+10=13.33)	VM3,VM9

2.5 MAX-MIN Algorithm

MAX-MIN sort tasks based on completion time; long tasks that take more completion time have the highest priority and are then given to the VM with the shortest overall execution time in the VMs list.

2.5.1 Advantages and Disadvantages of MAX-MIN

Advantages

- It maximizes the use of available resources.
- In terms of the performance, this technique outperforms the SJF, FCFS, and MIN-MIN algorithms.

Disadvantages

- It increases waiting time to small and medium tasks. For example, if there are six long tasks, in MAX-MIN scheduling algorithm they will take priority in six VMs in VM list, and short tasks must be waiting until the large tasks finish.

2.5.2 Example Calculations of MAX-MIN

When using MAX-MIN, the following work mechanism will be used:

Table 2.5 shows how the tasks described in Table 2.1 are sorted from largest task to smallest task based on highest completion time.

Table 2.5 A set of tasks sorted based on MAX-MIN scheduling algorithm

Task	Length
T1	100000
T8	100000
T7	90000
T15	80000
T2	70000
T14	25000
T13	20000
T9	15000
T6	10000
T3	5000
T12	4000
T5	3000
T11	2000
T4	1000
T10	1000

The tasks are then allocated to the VMs in the VMs list that have the minimum execution times. The tasks assigned to the VMs list sequentially are shown in the following:

$$VM6 = \{T1 \longrightarrow T13 \longrightarrow T11\}$$

$$VM5 = \{T8 \longrightarrow T9 \longrightarrow T10\}$$

$$VM4 = \{T7 \longrightarrow T6 \longrightarrow T4\}$$

$$VM3 = \{T15 \longrightarrow T3\}$$

$$VM2 = \{T2 \longrightarrow T12\}$$

$$VM1 = \{T14 \longrightarrow T5\}$$

According to Table 2.6, the small and medium tasks must wait in the task list until the large tasks are completed.

Table 2.6 Task waiting time in the MIX-MIN

Task	Length	Execution Time	Waiting Time		
T1	100000	40	VM6		
T8	100000	40	VM5		
T7	90000	60	VM4		
T15	80000	53.33	VM3		
T2	70000	140	VM2		
T14	25000	50	VM1		
T13	20000	8	Wait(40)	VM6	
T9	15000	6	Wait(40)	VM5	
T6	10000	6.66	Wait(60)	VM4	
T3	5000	3.33	Wait(53.33)	VM3	
T12	4000	8	Wait(140)	VM2	
T5	3000	6	Wait(50)	VM1	
T11	2000	0.8	Wait(48)		VM6
T4	1000	0.4	Wait(46)		VM5
T10	1000	0.67	Wait(66.67)		VM4

2.6 Chapter Summary

In this chapter, a detailed discussion of cloud technology is discussed. A detailed explanation of the essential properties of cloud computing, service models and deployment models are discussed. A detailed explanation of task scheduling is presented. Finally, examples of static scheduling algorithms are described.

CHAPTER-3

MULTI-OBJECTIVE TASK SCHEDULING USING K-MEANS ALGORITHM IN CLOUD COMPUTING

The cloud is an environment in which many factors must be optimized. This thesis focuses on cloud scheduling optimization using task clustering and task allocation policies. In this chapter, the software tools that support building the proposed system, Cloudlet Scheduling Policies, Task Clustering using K-Means Algorithm, and the detailed process of the proposed system are described.

3.1 Software Tools for Research

Since CloudSim is a Java-based simulation tool, it can be used with either the Eclipse or NetBeans IDEs. CloudSim provides several classes that assist the cloud computing simulation environment. In order to apply new scheduling algorithm, it is necessary to understand existing allocation policies and the classes that support these allocation strategies. Because clustering is a novel concept in CloudSim, various new classes have been defined. In the proposed system, the Eclipse IDE is used for application design and development.

3.1.1 Java Development Kit (JDK)

Oracle Corporation creates it in the form of a binary product. JDK was designed for Java developers working on Solaris, Linux, Mac OS X, or Windows. The JDK includes a private JVM and a few other resources for completing the Java application development.

3.1.2 CloudSim

CloudSim can be used to test the proposed algorithm's correctness. It was used to test task grouping and scheduling in a cloud computing simulation, which was based on the Java programming language and allowed both system and modeling of cloud computing system components. [5].

CloudSim simulator is used for cloud computing environment modification and simulation. It provides classes for representing data centers, virtual machines, applications, users, computing resources, and rules for managing various system components (e.g., scheduling and provisioning).The advantages of CloudSim are as follows: [6]:

- Testing services in a repeatable and controlled environment is free-using (no cost).
- The performance is testing the bottleneck before deploying on actual cloud.

As illustrated in Figure-3.1, the layered is patterned of the CloudSim tool model and architectural parts, with the upper layer to the user code exposing fundamental entities for hosts (number of machines, their specifications, and so on), applications (number of tasks and their requirements), VMs, number of users and its application types, and broker scheduling policies [7].

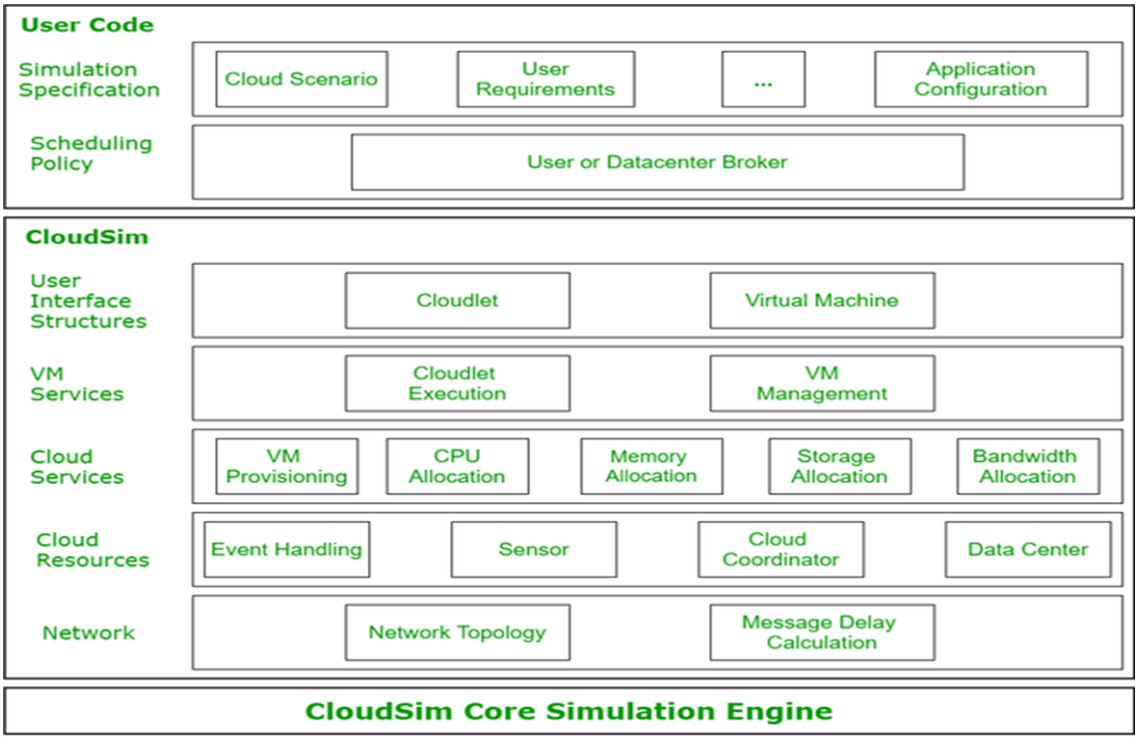

Figure-3.1 CloudSim Architecture

This thesis is simulated on CloudSim library. CloudSim is a simulation tool which provides all the features of the cloud. It provides essential classes for describing data centers, resources, virtual machines etc.

- **Datacenter:** This is a collection of hosts that are in charge of managing virtual machines throughout their life cycles. It provides resources to providers in a cloud computing environment (memory, cores, capacity, and storage). The essential infrastructure level services (hardware, software) are represented by this class. The datacenter component implements a set of policies for allocating bandwidth, memory, and storage devices, as shown in Figure 3-2:

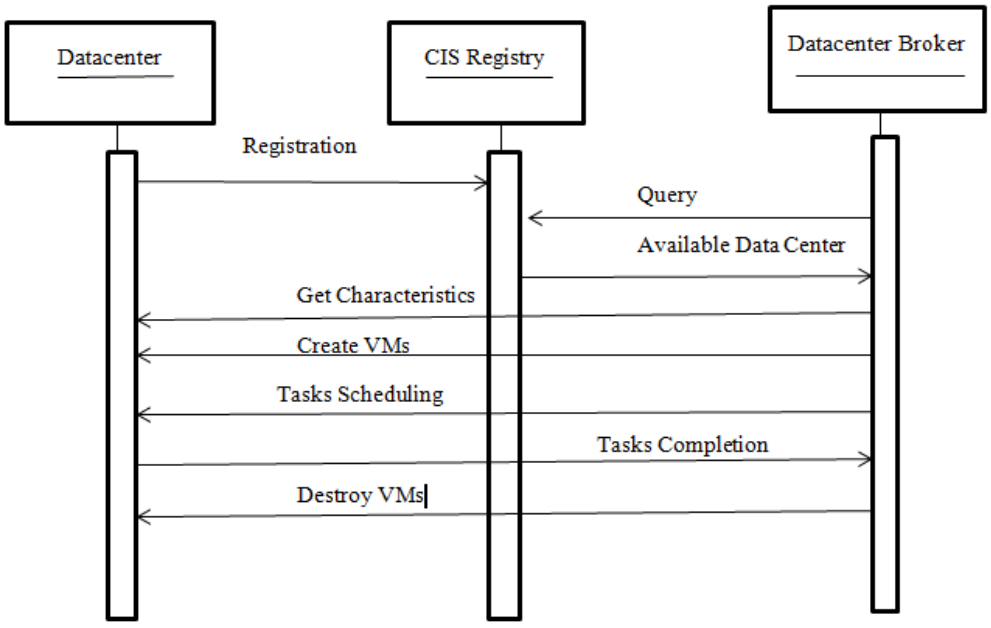

Figure-3.2 Simulation Data Flow

- **Cloud Information Services (CIS):** It is a cloud computing service that provides information on all computing resource registrations. CIS collects cloud computing resource information such as operating system (Windows, Linux), management policy (time share, space sharing), resource index, and MIPS processor capabilities. The addition to CIS gives the user information about the resources' availability.

- **Datacenter Broker**: It is a service provider that distributes service tasks across clouds based on user QoS needs. The cloud broker makes a request for QoS to the cloud service provider, and the list of VMs is created on the VM server. Here tasks priority is assigned according to the QoS value of task [7].
- **Host**: In the cloud, it represents a physical computing node. A pre-configured processing capability (measured in MIPS), memory, storage, and a scheduling policy for allocating processor cores to virtual machines are all created [6].

The researcher is using CloudSim as a framework for modeling and simulation of cloud computing infrastructures and services, based on its core functions. It is based on the Cloud Computing and Distributed Systems Laboratory in particular. The researcher will be able to test its scenarios and configurations, enabling for the creation of best practices in all aspects of cloud computing. Figure 3.3 depicts CloudSim features.

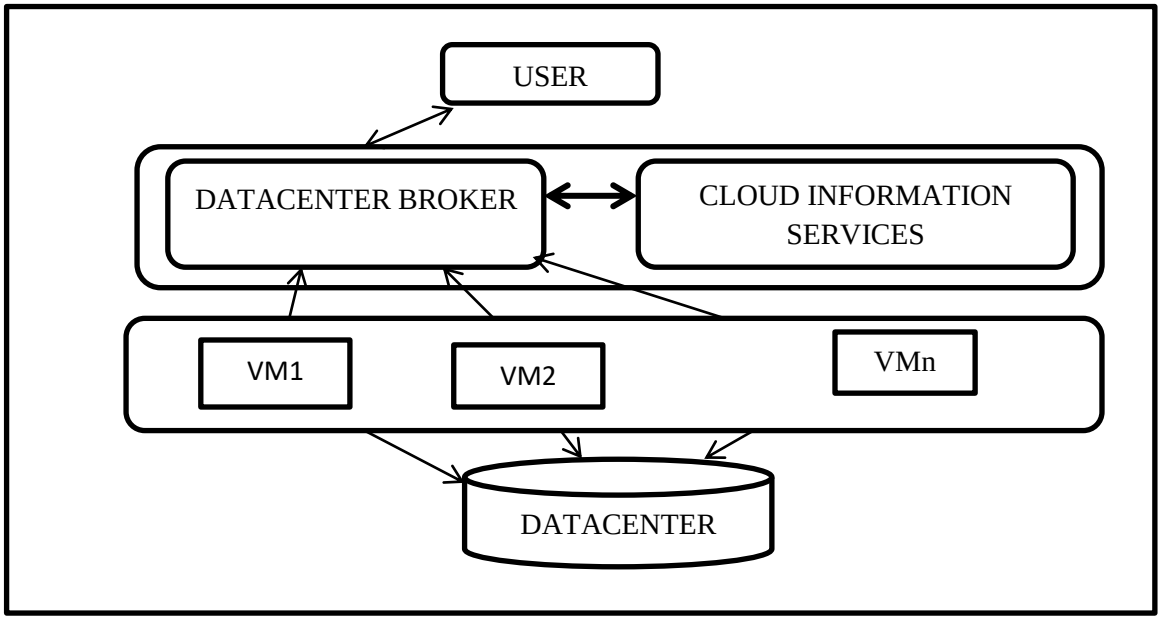

Figure-3.3 Features of CloudSim

3.2 Scheduling Policies

The scheduling scenario has been effectively addressed by the CloudSim simulation toolkit framework, which has built it as a collection of programmable class hierarchies with parent classes as:

- VmScheduler
- CloudletScheduler

In addition, researchers utilizing the CloudSim simulation toolkit have simulated Virtual Machine (VM) and Task (Cloudlet) scheduling as one of the most essential and popular use cases. Figure 3.4 describes this case [7].

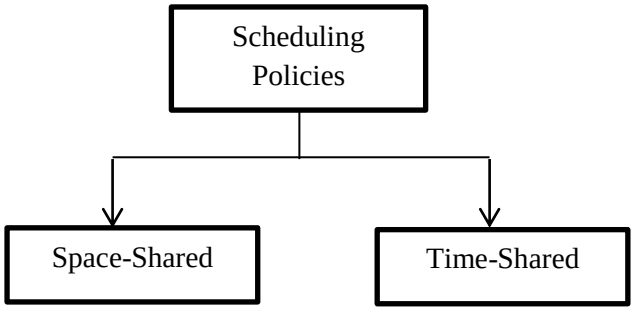

Figure-3.4 Scheduling Policies

3.2.1 Virtual Machine Scheduling in CloudSim

The VmScheduler class is an abstract class that defines and implements the policies for sharing processing power among virtual machines on a given host. The "org.cloudbus.cloudsim" package of CloudSim contains these classes. The following sorts of policies implemented as classes are included in the description of this abstract class:

- **VmSchedulerTimeShared**: This class implements the VM scheduling policy, which assigns one or more processing elements to a single virtual machine and allows several virtual machines to share processing components within a given time slice. In policy definition, this class additionally incorporates the overhead of VM allocation switching (similar to context switching). If the specified number of processing elements is not available, the VM allocation will fail. If the VM requests a quad-core processor but the allocated host only has a dual-core processor, the allocation will fail.
- **VmSchedulerSpaceShared**: This class implements the VM scheduling policy, which allocates one or more processing elements to a single virtual machine, but

does not permit processing element sharing (i.e., the assigned VM will use all of the required resources until the VM is destroyed). Furthermore, if any virtual machine demands a processing element that is not accessible at the time, the allocation will fail.

- **VmSchedulerTimeSharedOverSubscription**: This is an additional feature of the VMSchedulerTimeShared VM scheduling policy, which allows the virtual machine(s) to oversubscribe processing components (i.e., the scheduler allows the allocation of VMs that demand more CPU capacity than is available). As a result of the oversubscription, performance will decrease.

3.2.2 Cloudlet Scheduling in CloudSim

The "CloudletScheduler" is an abstract class that defines the fundamental skeleton for implementing a policy for cloudlet scheduling on a virtual machine. The "org.cloudbus.cloudsim" package of CloudSim contains these classes. This abstract class's definition is expanded to include the following sorts of policies implemented as classes:

- **CloudletSchedulerSpaceShared:** This class implements a scheduling strategy for virtual machines to run cloudlet(s) in a space shared environment (i.e., only one cloudlet will be executed at a time). It means that cloudlets are queued together and requests are handled one at a time by each computing core. Batch processing is related to space-sharing.
- **CloudletSchedulerTimeShared**: This class implements a cloudlet scheduling policy allowing Virtual machines to execute cloudlets in a time-shared environment (i.e., more than one cloudlet will be submitted to the virtual machine, with each receiving its own share of time). It means that numerous requests (cloudlets) are processed at the same time, but they must share the virtual machine's computational capacity (via simulating context switching), affecting each other's processing time. In CloudSim, it basically impacts the completion time of a cloudlet. Time-sharing, often known as round-robin scheduling, refers to the concept of sharing executing power (such as CPU, logical processor, and GPU).

- **CloudletSchedulerDynamicWorkload:** This implements a particular policy of virtual machine scheduling assuming that there is just one cloudlet running as an online service with a different workload need based on the need for peak/offpeak user load at a specific moment.

This system focus on **CloudletSchedulerSpaceShared** is used for task scheduling. In space-shared scheduling policy, only one task is allowed to be executed at a given instance of a time on VM.

3.3 Overview of Task Scheduling

The process of ordering incoming requests (tasks) in a specified order to maximize the utilization of available resources is known as task scheduling. Because cloud computing is a technology that provides services via the Internet, service users must submit their requests online. The main benefit of cloud computing is that it encourages good resource utilization [19]. Task scheduling and resource allocation are thus two sides of the same coin. Each has an impact on the other.

3.3.1 Scheduling Levels in Cloud Computing

There are two levels of scheduling algorithms in cloud computing:

- First level: in host level where a set of policies to distribute VMs in host.
- Second level: in VM level where a set of policies to distribute tasks to VM.

This system focus on VM level to schedule tasks. A task/job scheduler is used to map tasks for execution to the assigned VMs at the VM level. It is known as Task Scheduling. A VM scheduler is used at the host level to allocate VMs to physical hardware. This is commonly referred to as VM scheduling. Tasks can be classed as independent or dependent on their task dependency. Independent tasks have no dependencies on other tasks and do not require a priority order to be followed during the scheduling process. However, dependent tasks have priority order based on task dependencies and must be followed during the scheduling process.Figure-3.5 shows the levels of scheduling.

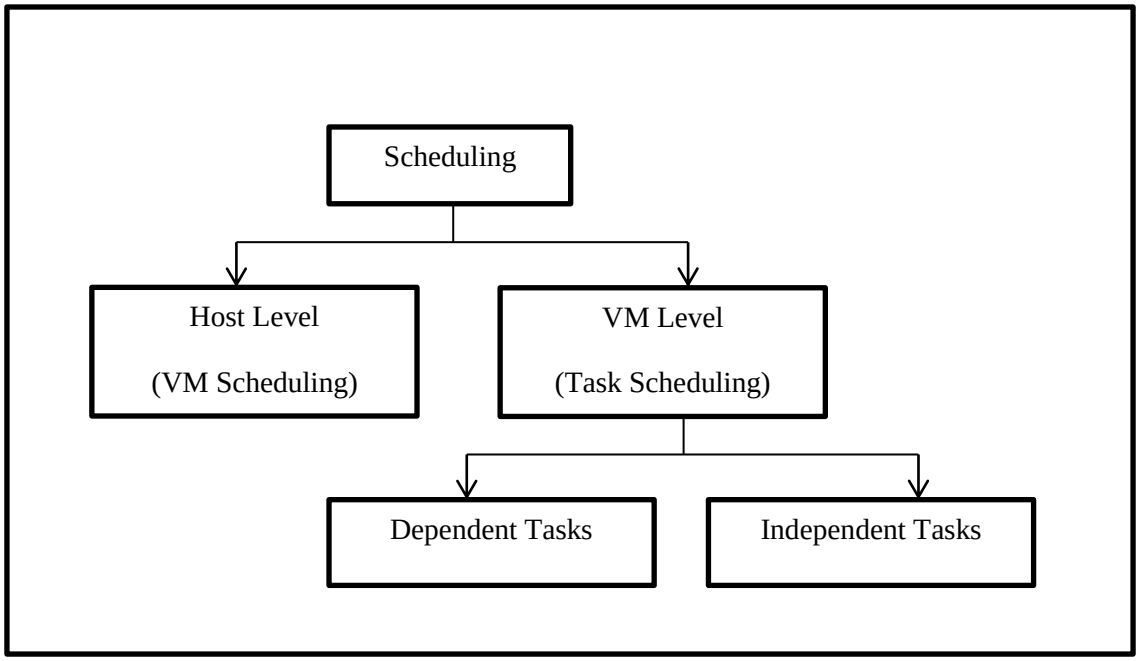

Figure-3.5 Scheduling Levels

VM scheduling is the allocation of Vms to run on the appropriate physical machines PMs to insure the implementation of tasks. This will improve the utilization of resources as well as managing the load balancing of all the systems.VM scheduling is important to ensure the Quality of Service (QoS) and Service Level Agreements (SLA) agreed by the cloud service providers and customers.

3.4 Task Clustering using K-Means

Data clustering [18] is to split the elements in a dataset into subsets where elements of the same group are more similar to each other than the elements from different groups. Various clustering techniques are available in data mining, among which K-Means clustering is a simple and efficient approach. Compared to the other data clustering algorithms is an important and widely used technique in data analysis and data mining. The prime objective, K-Means clustering works efficiently in handling datasets with a single attribute. It is also quick and simple to implement. Hence, the proposed system makes use of K-Means to form task clusters to reduce makespan and perform a fair distribution of workload among the VMs.

3.4.1 Steps of the Proposed System

The essential operations that are performed in the proposed task scheduling algorithm are clearly explained**.** First of all, a list of unscheduled tasks is accepted**.** Following the creation of tasks clusters are formed using the K-means algorithm. Then sort the clusters in order of decreasing size (the higher the centroid, the larger the cluster). Finally, tasks are mapped to the best VM using the Resource Allocation Algorithm. The overview of the proposed system is illustrated in Figure-3.6.

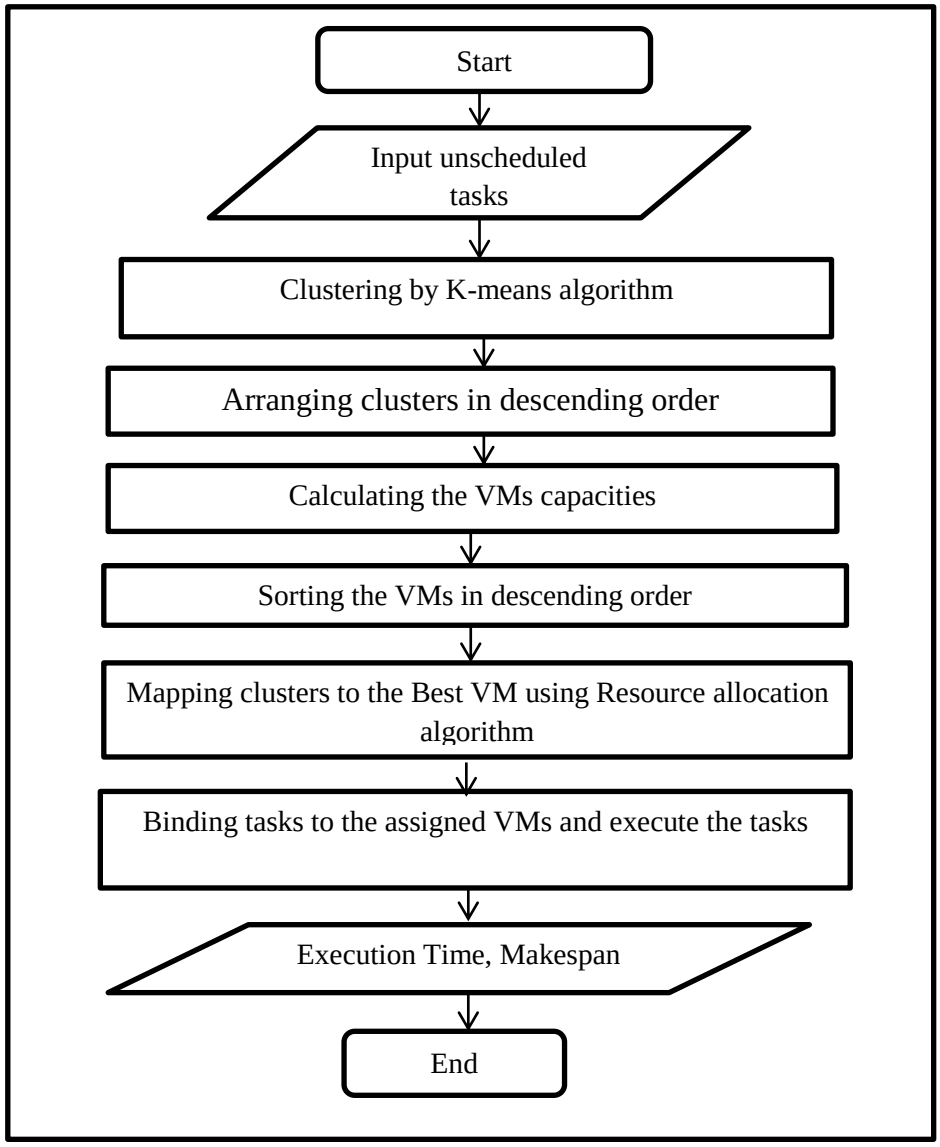

Figure-3.6 Overview of Proposed System

CHAPTER 4
SYSTEM IMPLEMENTATION AND EXPERIMENTAL RESULTS

The implementation details and experimental results are described in this chapter. In the system implementation, the dataset and experimental environment are described. The proposed system is then compared to FCFS.

4.1 Proposed System Architecture

The process used to choose the resources to execute tasks in order to reduce waiting and execution time is referred to as task scheduling algorithms. Traditional scheduling techniques cannot fulfill the demands of cloud computing due to its novelty. Various cloud-based task scheduling algorithms such as Shortest-Job First, Round Robin, and First Comes First Serves which are single-criteria based algorithms. So, in order to satisfy more than one criterion, multi-objectives based task scheduling algorithms are needed.

Clustering-based Task Scheduling algorithms can give a new solution for the above problem as they can be used for scheduling large dataset of cloudlets (tasks) with more than one criteria (Task length and Deadline) easily and effectively. In this system, task groupings are created using the K-mean clustering technique. The suggested multi-objectives task scheduling algorithm for cloud computing is an optimal task scheduling algorithm with the minimum total execution time. Figure 4.1 depicts the system architecture.

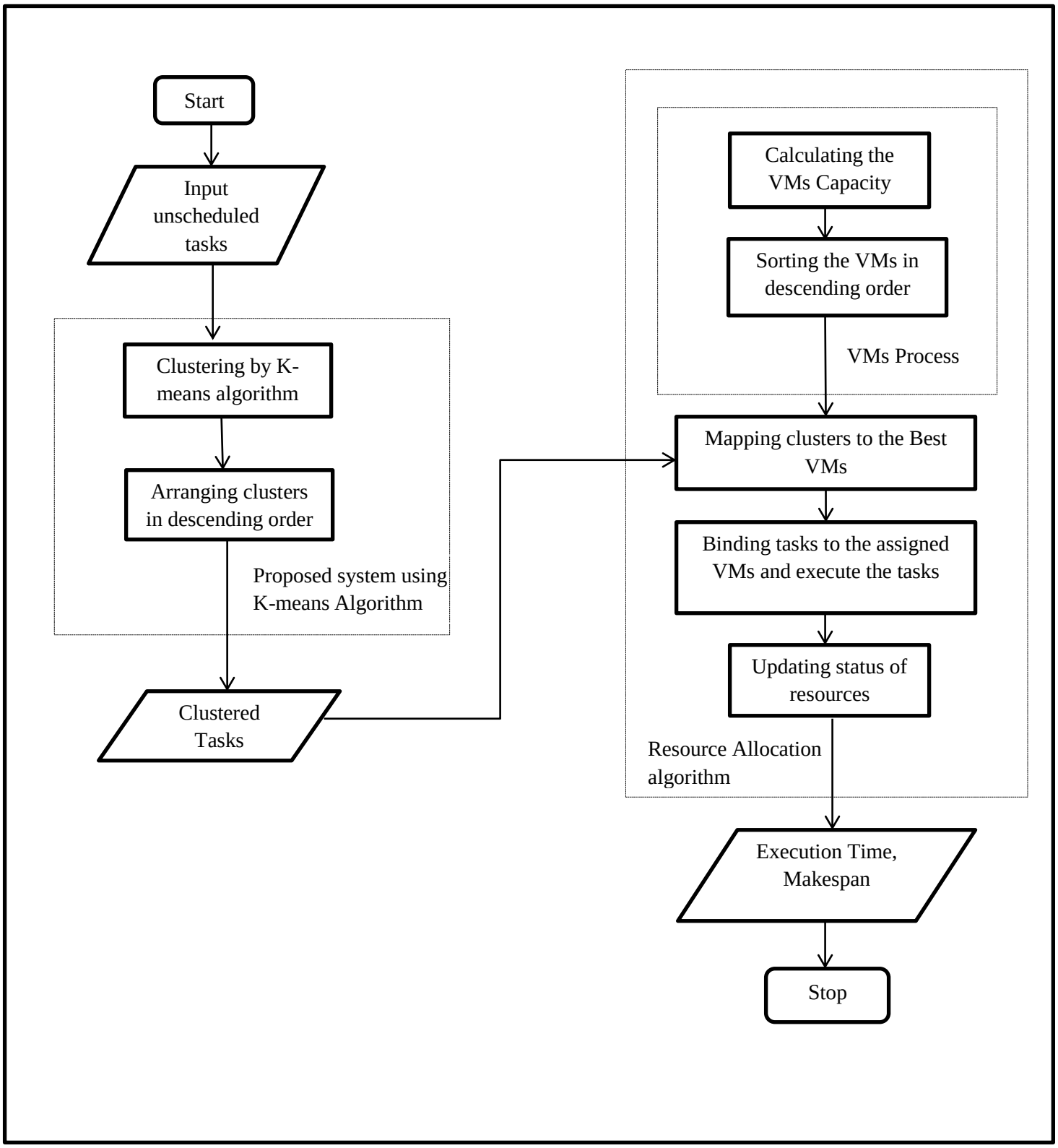

Figure-4.1 Proposed System Architecture

4.1.1 Multi-objective Task Scheduling using K-means Algorithm

Input: List of unscheduled Tasks (Cloudlets) Set

Output: Task Clusters

Start

Step1: Select k points according to number of VMs

Step2: Form K clusters of cloudlets by assigning each point to its closest centroid.

Step3: Calculate the centroid using Euclidean distance of all the task clusters

$$\mathbf{dist((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}}$$

Where x= cloudlet length

y=deadline

Step4: For each task cluster, find the mean as

$$\mu_k = \frac{1}{N_k} \sum_{q=1}^{N_k} x_q$$

where N_k = the number of instances belonging to cluster k

μ_k = the mean of the cluster k.

Step5: Recomputed the centroid for each cluster.

Step6: Repeat Step2 to Step5 Until centroid do not change.

Step7: Arrange clusters in descending order

Step8: Map clusters to the Best VM using Resource Allocation algorithm.

End

4.1.2 Resource Allocation Algorithm

Input : Task (Cloudlet) Clusters
: List of Virtual Machines (VMs)

Output: Allocated tasks to VMs

Algorithm:

Start

Step1: Set i=0

Step2: **While** VM[i] is not empty **do**

Step3: Calculate the capacities of VM[i] using equation (1),

$$C_i = Pro_{ni} * P_{mips} + VM_{bwi} \quad \textbf{(1)}$$

Where Pro_{ni} is the number of processors in VM_i

P_{mips} is millions of instructions per second of all processors in VM_i

VM_{bwi} is the communication bandwidth ability of VM_i

Step4: Set i=i+1

Step5: **End While**

Step6: Sort the VMs in **descending order** based on their maximum capacity

Step7: Set j =0

Step8: **While** cluster [j] is not empty **do**

Step9: Assign cluster[j] to VM and execute it

Step10: Update status of resources.

Step11: Set j=j+1

Step12: **End While**

End

4.2 Dataset used in Thesis

Different tasks of parameter values in thesis experiments are implemented using the CloudSim tool (dataset). GoCJ was used to generate the dataset (about 1000 tasks). The experiments were carried out using a Google cluster workload. Analyzing the Google cluster workload traces, a realistic Google-like workload is generated using Monte-Carlo simulation [3]. In this system, task sizes range from 15,000 MI to 900,000 MI. The sample dataset used in this system is shown in Table 4.1.

Table 4.1 Google cluster workload

Length	I/o File Size	Output file size	Deadline
83000	324	1139	83
91000	337	943	91
95000	382	489	95
27500	1050	514	27.5
49000	1276	748	49
81000	532	561	81
47000	1008	940	47
95000	963	986	95
40000	1065	785	40
103000	1024	429	103
55000	484	679	55
99000	1255	485	99
71000	965	1127	71
125000	942	1083	125
111000	1002	1119	111
123000	1128	430	123
87000	884	1030	87
51000	447	1079	51

4.3 Experimental Evaluation

Eclipse (Oxygen) was used to develop the implementation code for the proposed task scheduling method. This code is used to extract results that are utilized to evaluate the suggested task scheduling algorithm's performance. A 64-bit Windows 10 operating system with an i3 processor and 8 GB of RAM serves as the simulation environment. Table 4.1 lists the simulation parameters used to implement the algorithm.

Table 4.2 Simulation Parameters

Entity	Parameters	Values
Cloudlet	Number of Cloudlets	1000
	Length	15000-900000MIs
Host	Number of Hosts	2
	Type of Policy	Time Share
	RAM	2048MB,3096MB
	Storage	1TB
	Bandwidth	10GB
Virtual Machine	Number of VMs	5
	MIPS	250-3000MIPS
	Memory Capacity of VMs	512-4196MB
	Bandwidth	1000MB
	Type of Policy	Space Share
	Number of CPUs	1-2
	VMM	Xen
	Operating System	Linux
	Number of Task Clusters(k)	5

4.3.1 Performance Measures

A performance metric is a commonly accepted definition of a measurable quantity that represents some aspect of performance. It should be measurable and consistent with the performance goals of the scheduling problem. Any solution to the scheduling problem should try to minimize the makespan and execution time of the tasks. Hence, the proposed method is evaluated using these metrics.

Execution Time: It is the amount of time taken by a VM to run or execute a task. The average execution time defined in equations (4.1), (4.2) and (4.3) are used to measure the performance of the datacenter.

$$ET_{ij} = \frac{T_{length(i,j)}}{VM_{comp(i,j)}} \qquad \text{Equation (4.1)}$$

$$VM_{comp(i,j)} = VM_{pesnum(i,j)} * VM_{mips\ (i,j)} \qquad \text{Equation (4.2)}$$

$$\overline{ET} = \frac{\sum_{j=1}^{m} \sum_{i=1}^{n} ET_{ij}}{n}$$ Equation (4.3)

where ET_{ij}, represents the execution time of task i on virtual machine j

$T_{length(i,j),}$ represents tasks instruction length

$VM_{comp(i,j),}$ represents the calculation ability of the j virtual machine

$VM_{pesnum(i,j)}$, indicates the number of CPU virtual machine j

n represents the total number of tasks

Makespan: Makespan is defined as the maximum time taken by a VM to complete the tasks in the task queue. It is denoted as the maximum of the completion time of all the tasks which is given by equation (4.4).

MCT = max $\{CT_{ij} |$ i$\in$ 1, 2, 3, .., m, j $\in$ 1, 2, 3... n$\}$ Equation (4.4)

where CT_{ij} , represents the time taken by VM j to complete task i

4.4 Experimental Results

The experiments are conducted by keeping the VM count as 5 and the task count as 500, 600, 800, and 1000. Here, both the tasks and the resources exhibit heterogeneous characteristics. Table 4.2 and Table 4.3 show the example of task clusters (tasks 1000) along with the task length, deadline, and descending order of each cluster.

Table 4.3 Average Calculation for Tasks (Cloudlet)

Tasks	Average Length	Average Deadline
Cluster0 (204)	88705	88.70588235
Cluster1 (243)	92364	92.36419753
Cluster2 (60)	1450819	737.5
Cluster4 (264)	92140.15152	92.14015152
Cluster3 (229)	89722.70742	89.72270742
1000		

Table 4.4 Descending order for Clusters

Before Sorting	After Sorting
Cluster0 (204)	Cluster2 (60)
Cluster1 (243)	Cluster1 (243)
Cluster2 (60)	Cluster4 (264)
Cluster4 (264)	Cluster3 (229)
Cluster3 (229)	Cluster0 (204)
1000	**1000**

In the proposed system, multi-objective task scheduling is performed with the aim of minimizing the makespan and execution time. The first comparison is performed between the traditional algorithm (FCFS) and the proposed algorithm based on calculated average execution time, and the second comparison is performed between the traditional algorithm (FCFS) and the proposed algorithm based on calculated makespan. The traditional scheduling algorithm (FCFS) and the proposed algorithm have been implemented on the same dataset. First, average execution time and makespan are measured, and the resultant values are given in Table 4.4 and Table 4.5 for 5 VMs. When compared to FCFS, the proposed algorithm reduces average execution time by 4%, 9%, 14%, and 24%, respectively.

Table 4.5 Average Execution Time Comparison for 5VMs with difference Task size

Execution Time (milli sec)		
Number of Tasks	**FCFS**	**Proposed System**
500	187.7291795	180.0403048
600	197.1855512	181.2537222
800	216.116459	189.12425
1000	248.973795	201

Table 4.6 Makespan Comparison for 5VMs with difference Task size

Makespan (milli sec)		
Number of Tasks	**FCFS**	**Proposed System**
500	1187.5	1161.401099
600	1283.332	1185.43956
800	1500	1257.85714
1000	1649.9994	1353.571429

After task clustering, tasks with large clusters are scheduled on high-capacity VMs, whereas tasks with smaller clusters are scheduled on low-capacity VMs. This reduces the average execution time which in turn reduces the makespan. The average execution time is reduced as a result, and the makespan is reduced. In comparison to FCFS, the suggested algorithm reduces the makespan by 2%, 8%, 19%, and 22%, respectively.

4.4.1 Comparison results between FCFS and proposed algorithm

The performance of this task scheduling system is evaluated using two metrics in this thesis. The researcher has been used Execution Time (ET) and Makespan measurements to evaluate the performance of the proposed system. The values of average execution time these algorithms have been registered in Figure-4.3 and Figure -4.4 by showing line graph and bar chart.

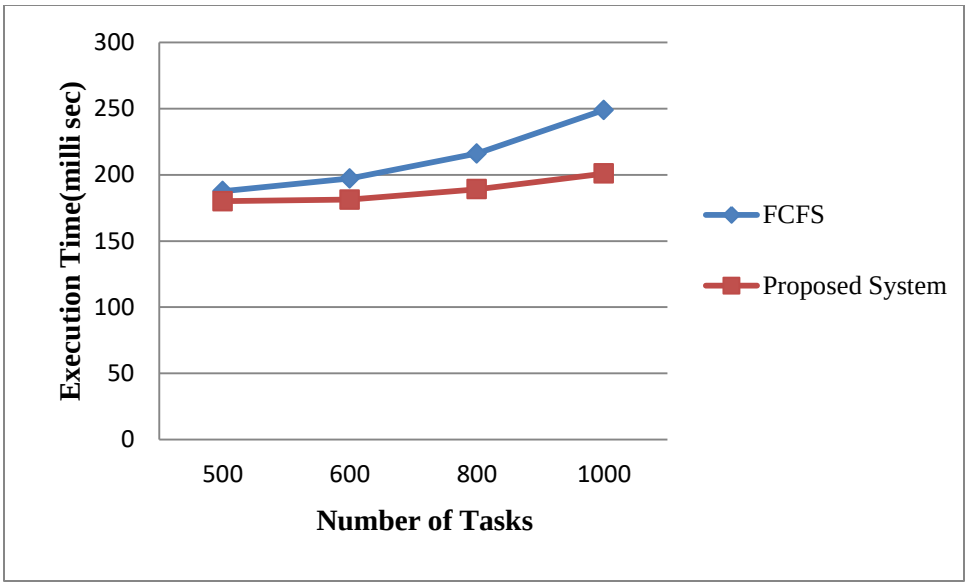

Figure-4.2 Average Execution Time using Line Graph

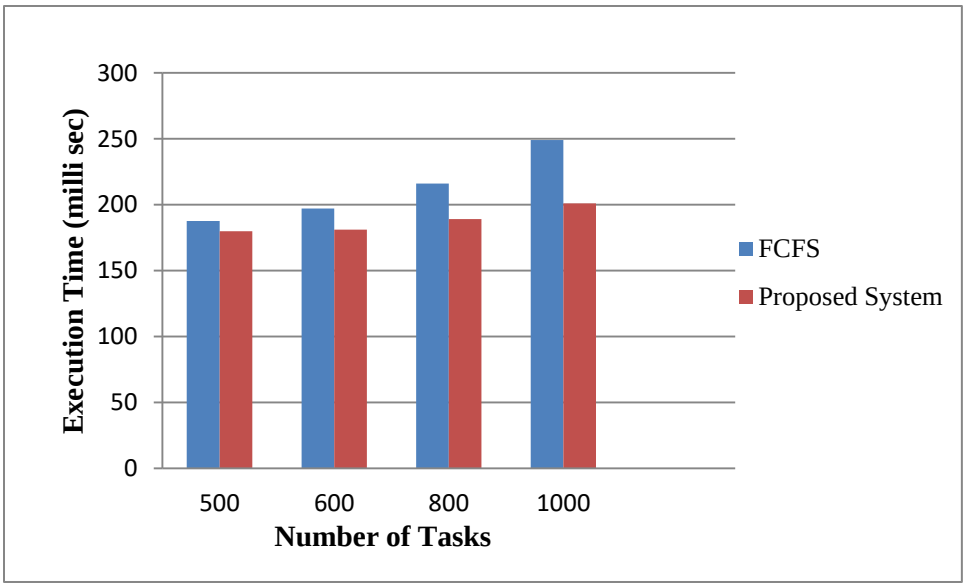

Figure-4.3 Average Execution Time using Bar Chart

Figures (4.5 and 4.6) show the comparison between the recorded values of makespan in both FCFS and proposed algorithms. These results show the proposed algorithm succeeded to achieve minimum execution time and makespan .Execution time is directly proportional to the Makespan in task scheduling.

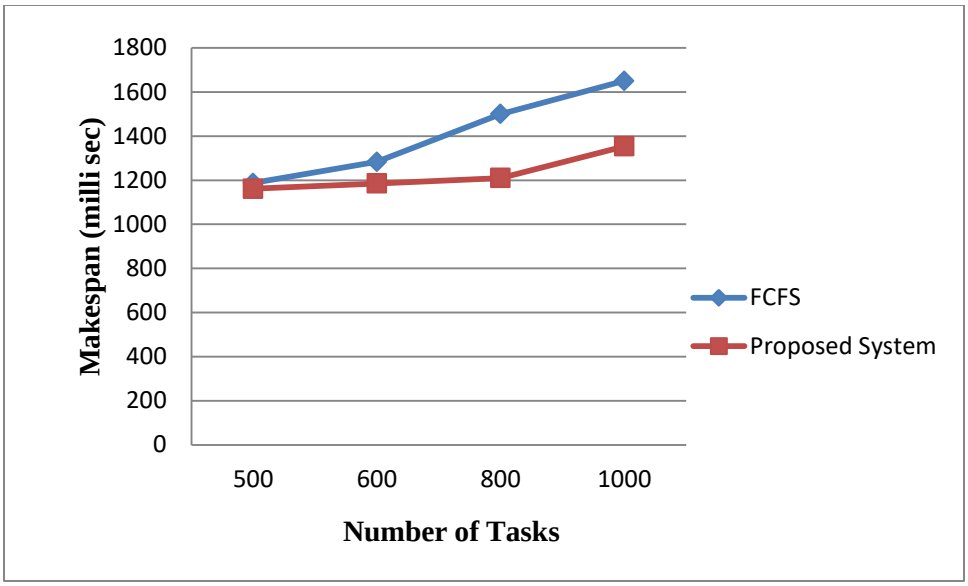

Figure-4.4 Makespan using Line Graph

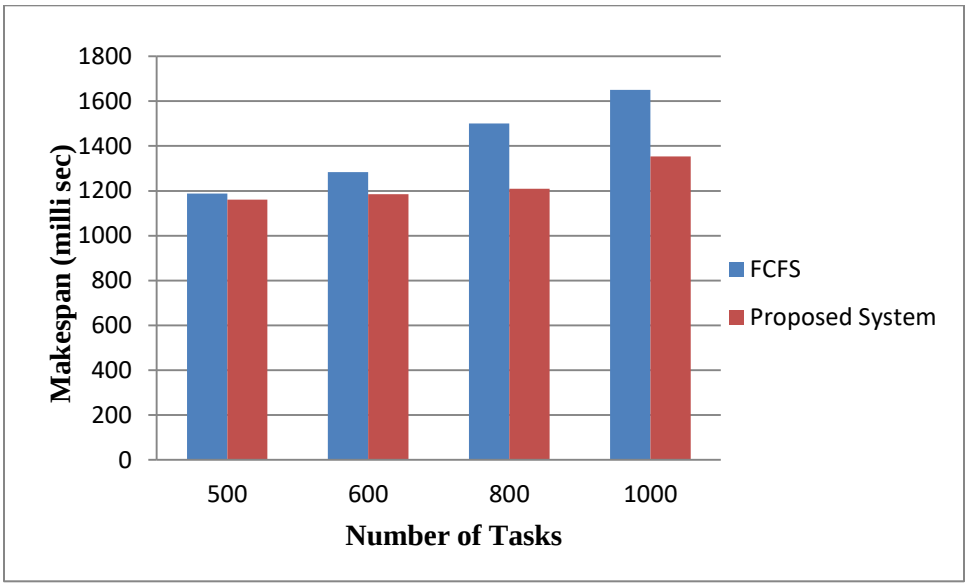

Figure-4.5 Makespan using Bar Chart

Above experiments conclude that multi-objective tasks scheduling algorithm is proposed which is integrated with K-means clustering algorithm for assigning tasks to VMs better outcomes than existing task scheduling algorithm (FCFS).When a big number of jobs from the different level of users are submitted to be executed in this system. The proposed algorithm has an ability to change easily the way of ordering task clusters for execution suitable VMs.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

Cloud is a widely used technology in many organizations. Optimizing its performance is a major challenge nowadays. It provides numerous advantages to its users and enterprises. The proposed task scheduling algorithm has been implemented to reduce execution time, makespan, and to make better resource utilization. Task Scheduling cannot be done based on individual criteria, but it depends on several rules and regulations. The number of users and their requests use to vary dynamically in the cloud. As a result, a scheduling technique that takes this heterogeneity into consideration when assigning requests to appropriate resources is required in order to reduce task completion times. K-means clustering is used in the suggested multi-objective task scheduling technique for cloud computing. This algorithm allocates clusters to the best VMs by calculating the capacity of VMs. According to the results of the experiments, the proposed method improves datacenter performance by reducing the makespan and execution time.

5.1 Advantages of the System

By applying this method, users can know how to apply data mining concepts to cloud computing. Many tasks are used in this system, and users can understand that parallel computing is the simultaneous use of multiple computing resources. Many task scheduling algorithms use a single criterion for resource allocation; however, the researcher improved the performance by using multiple criteria (task length and deadline).

5.2 Limitations of the System

There are some limitations to this system. The first limitation is that it is only tested on five VMs. It can only be used in independent tasks, which is the second restriction. The host level doesn't have any effect on assigning the task clusters to suitable VMs that have the VM level.

5.3 Future Extensions

During this research, some points were recorded as potential future work. The proposed scheduling algorithm can be evaluated on dependent tasks, which will definitely impact on the system performance and result in different outcomes. As the future work, K-Means clustering works better in a cloud environment than the traditional algorithm; it may also be replaced with other accessible clustering approaches. And then, the number of VMs and tasks can be added more and more to apply this scheduling algorithm. This system can also add more QoS parameters for task clustering.

APPENDIX A

This system will start with the View form as shown in Figure A-1.Then,user can be entered the system via log in form. If the login is successful, the user can get the message box about “Login Successful”. This is shown in Figure A-2.

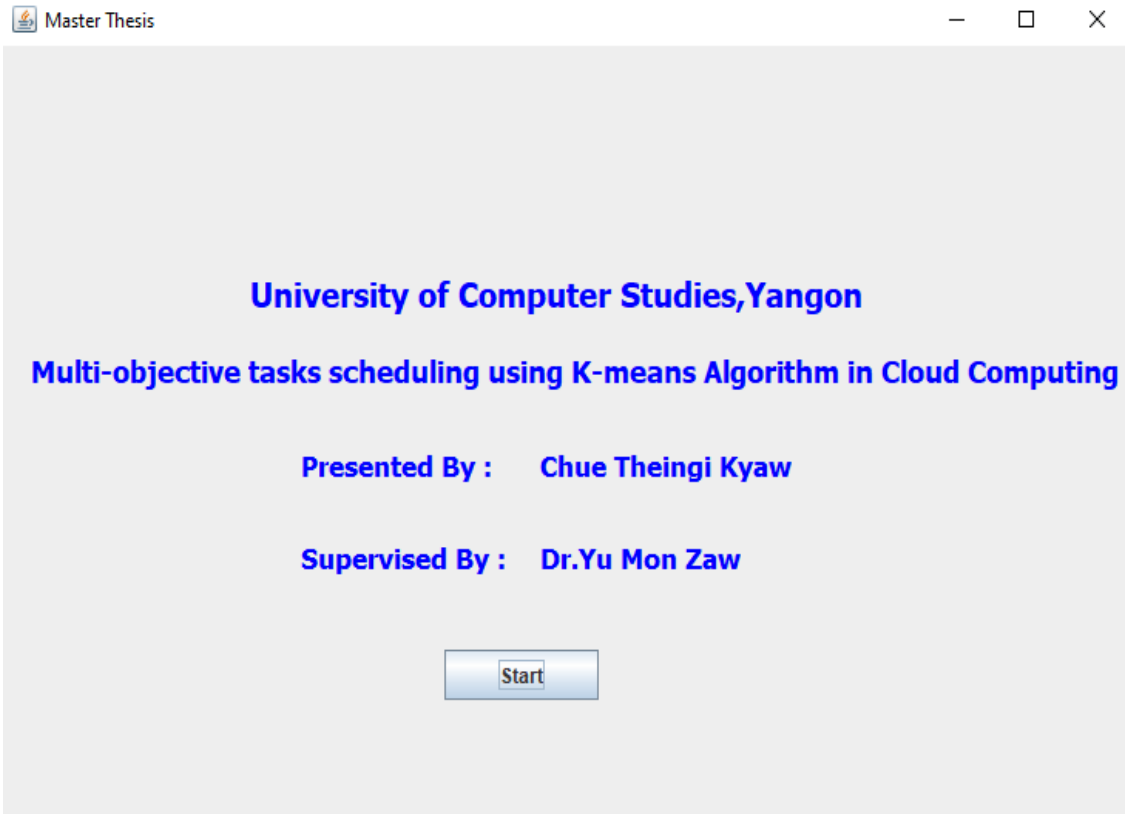

Figure A-1 View Form of the System

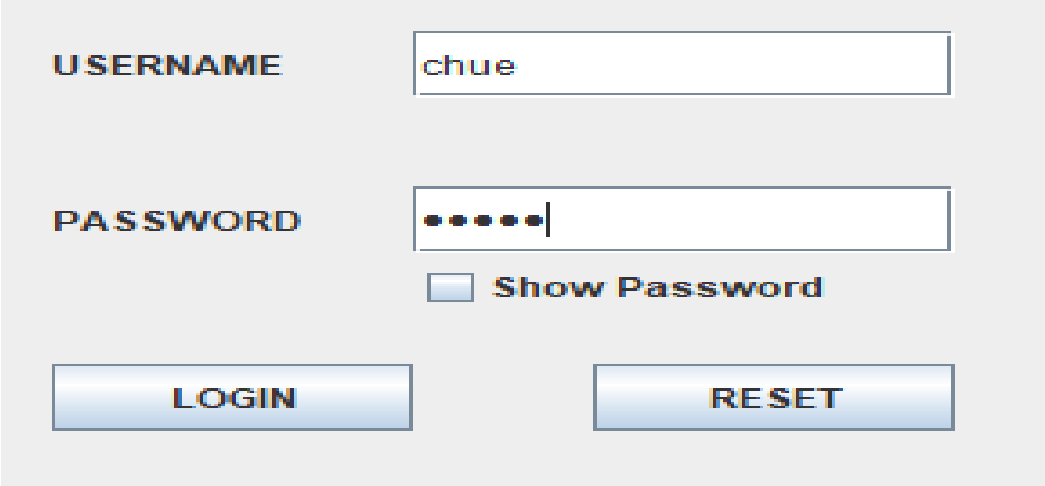

Figure A-2 Login Form of the System

When the user has log in to the system, he or she can see the main form of the system. The main form is about the description of the system. The user will see the message "Login Successful" if the login is successful. User can reset of login. When user click the first step of proposed system then user can view the upload form for cluster in Figure A-4.

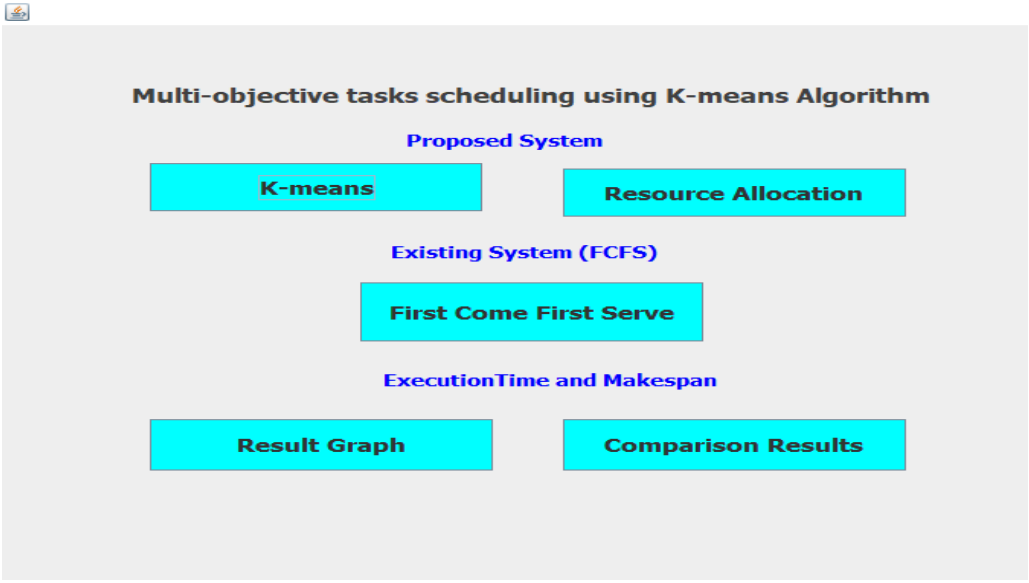

Figure A-3 Main Form of the System

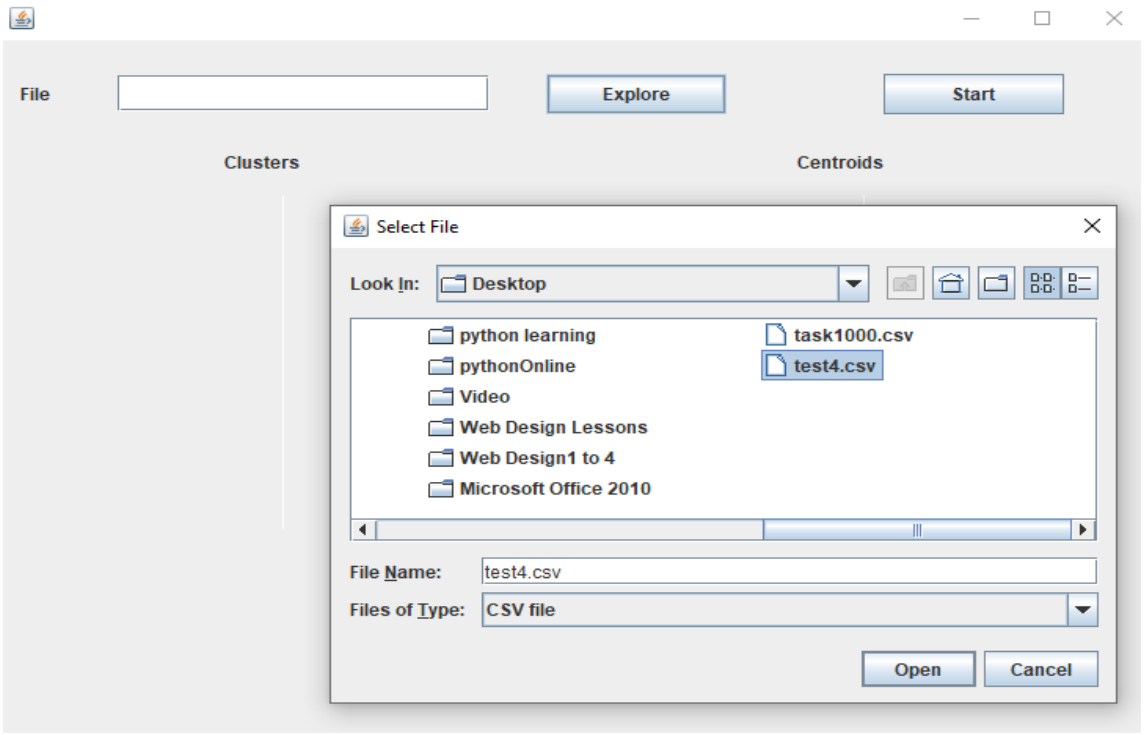

Figure A-4 File Upload Form

When user choose CSV file and click start button, the clustering method process and result the task clusters are appear in text box as shown in Figure A-5.

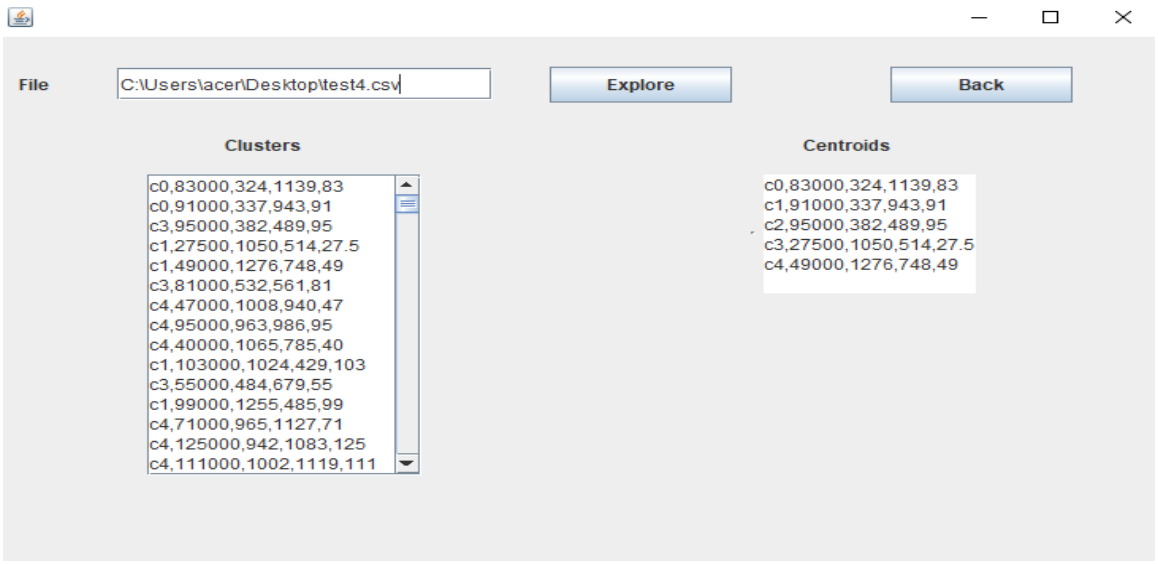

Figure A-5 Task Clusters Form

After click back button, user click another Resource Allocation button to apply the proposed system .When user click this button, the next form will appear to choose the VMs capacity for calculation maximum VMs capacity as shown in Figure A-6.

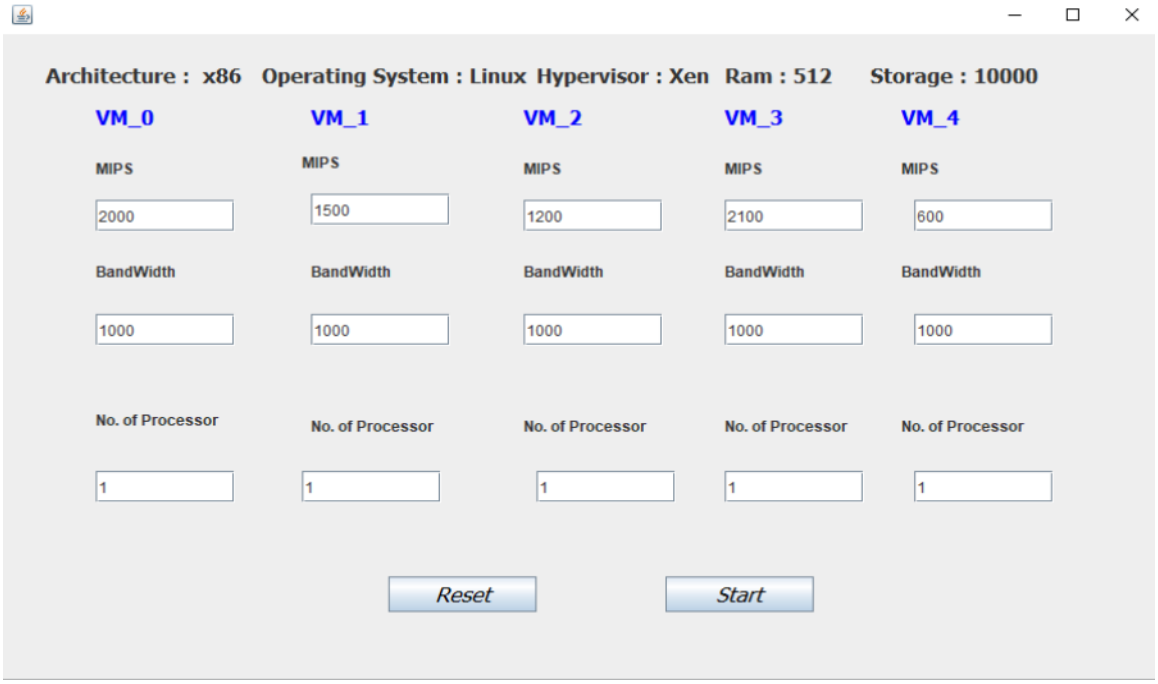

Figure A-6 Choose VMs capacity

When user clicks the next First Come First Serve button to apply the traditional algorithm, the above form will appear to choose the VMs capacity. Then pressing Result Graph, the user will be shown the current results of average execution time and makespan in both FCFS and proposed algorithms in Figure A-7, A-8.

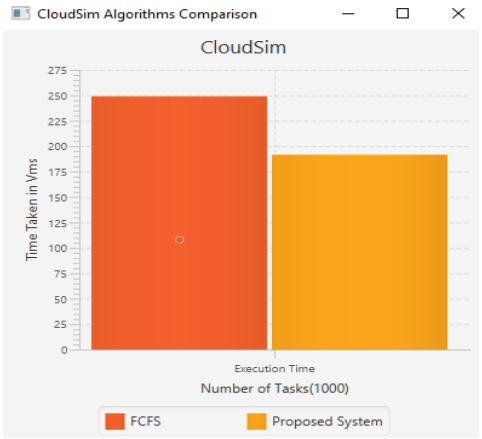

Figure A-7 Average Execution Time

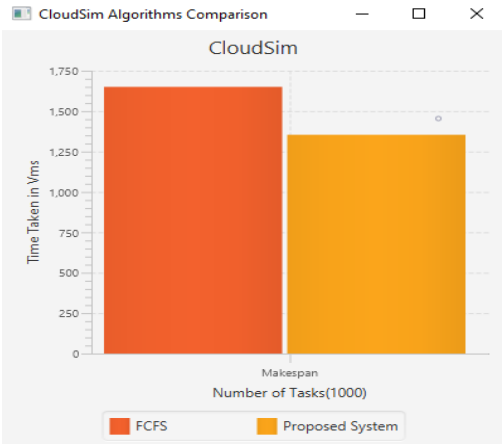

Figure A-8 Makespan

If user press Comparison Results button, Figures A-9 shows the comparison between the recorded values of Execution time and Makespan for all testing in this system.

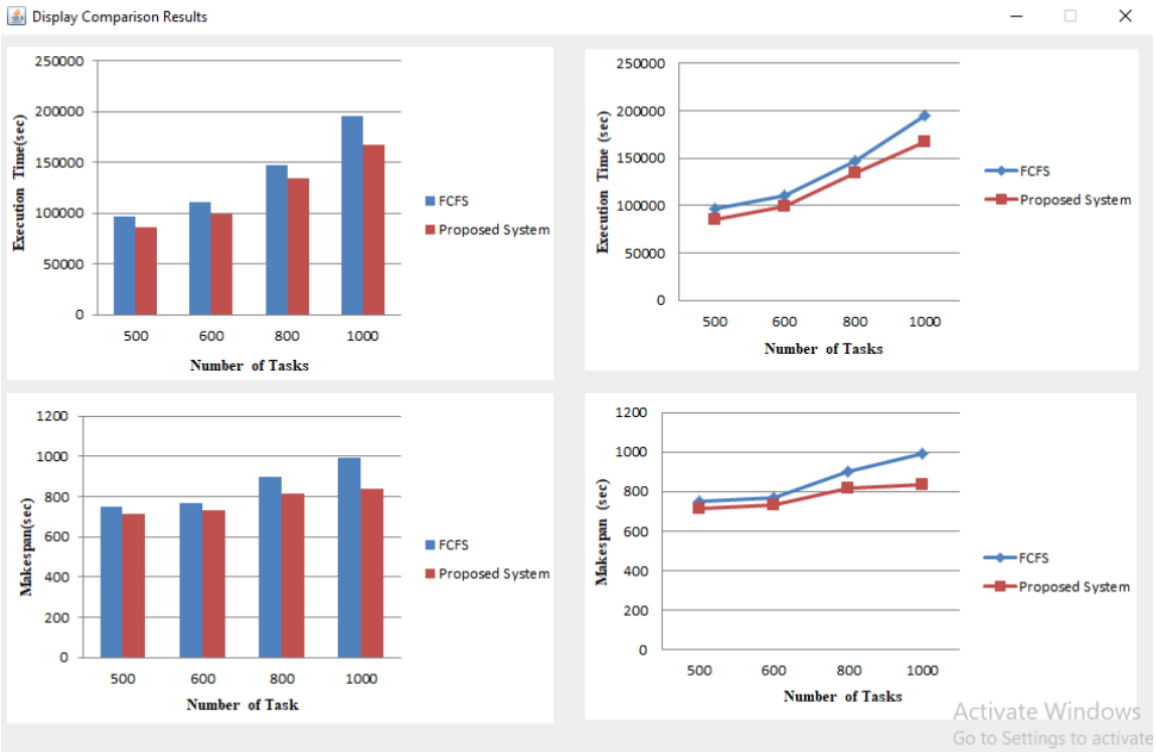

Figure A-9 Comparison Results for all testing

APPENDIX B

USING CLOUDSIM WITH ECLIPSE (IN WINDOWS MACHINE)

- Go to http://www.eclipse.org/downloads/to get Eclipse. .
- Eclipse can be extracted to a specific location. Let's use C:\eclipse as an example.
- Extract the CloudSim project to a specific location. Let's use C:\cloudsim3.0.3 as an example.
- Launch the Eclipse integrated development environment. To do so, navigate to C:\eclipse and launch the eclipse program.
- Select the workspace in Figure B-1, which is where Eclipse stores your projects.

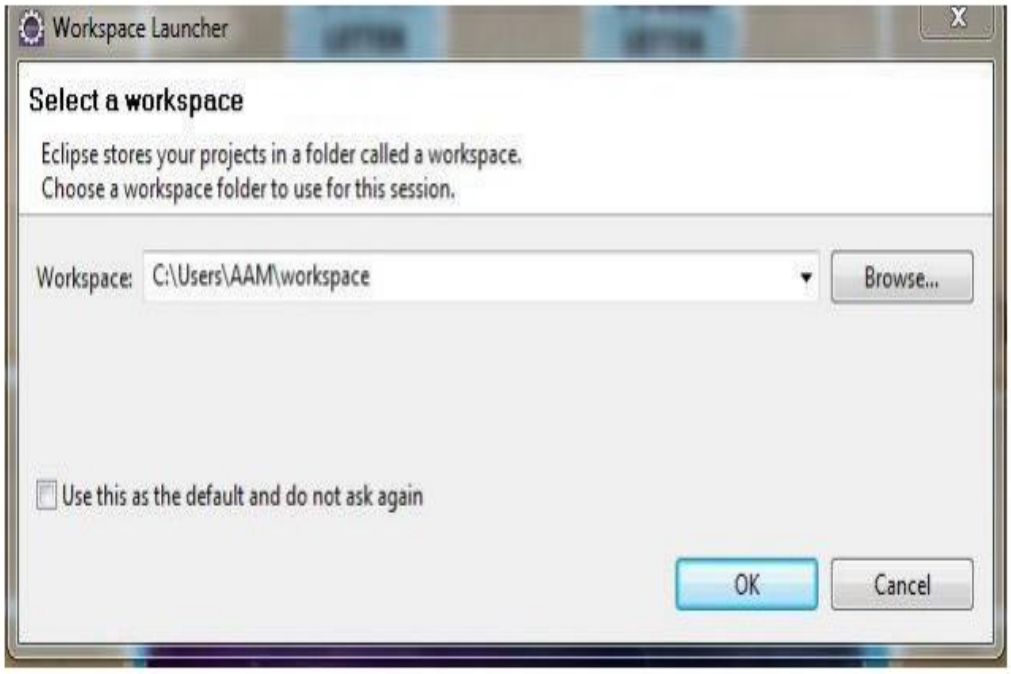

Figure B-1 Select Workspace

- As shown in Figure B-2, navigate to New Java Project in the Eclipse IDE.

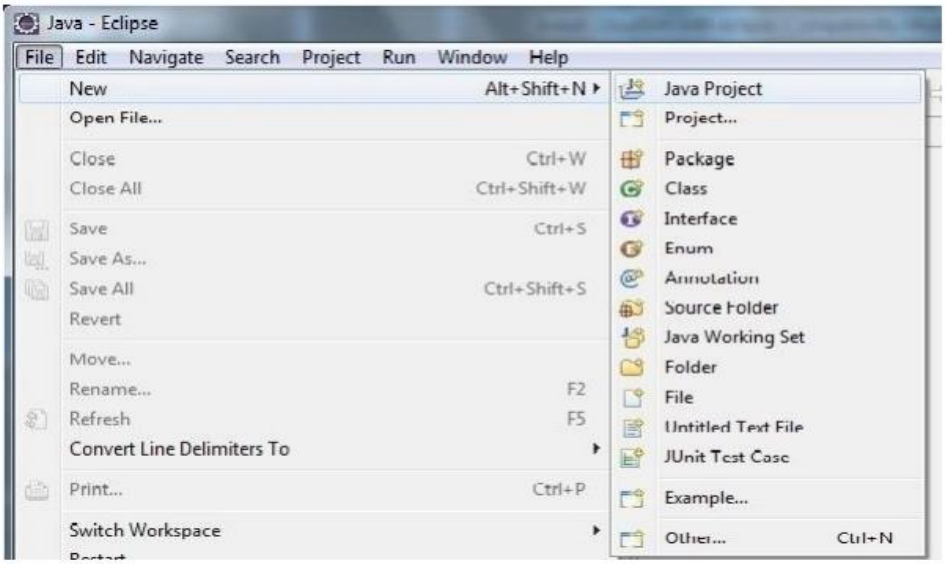

Figure B-2 New Java Project

- Use the project name as CloudSim as shown in Figure B-3, use the default location option and select the extracted CloudSim folder. And then, click "Finish".

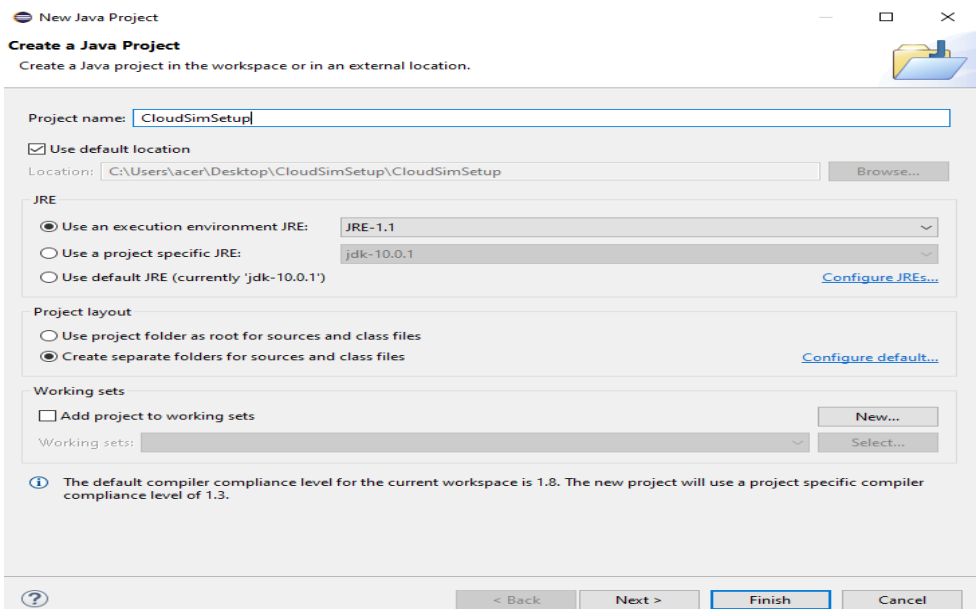

Figure B-3 Choose CloudSim folder

- Apart from the jars provided by the CloudSim library, below are the additional jars utilized by this project.

Apart from the jars provided by the CloudSim library, below are the additional jars utilized by this project.

  - Commons-math3-3.0.jar
  - Weka.jar
- Finally, CloudSim is installed in the Eclipse environment.

How to run the project

- There are two folders in the CloudSim package: examples and sources.
- This class is responsible for creating virtual machines, hosts, and datacenters:
  examples ->org.cloudbus.cloudsim.examples.power -> Helper.java
- In this work, clustering is performed as the first step of incoming tasks. It is performed in the class located at:
  examples ->org.cloudbus.cloudsim.examples->K-means.java
- Tasks are assigned to virtual machines in this class:
  sources -> org.cloudbus.cloudsim ->DatacenterBroker.java

AUTHOR'S PUBLICATIONS

[1] Chue Theingi Kyaw, Yu Mon Zaw, "Multi-Objective Task Scheduling using K-means Algorithm in Cloud Computing", Parallel & Soft Computing, University of Computer Studies, Yangon, 2022.

REFERENCES

[1] Awan, M., & Shah, M. A, A Survey on Task Scheduling Algorithms in Cloud Computing Environment , International Journal of Computer and Information Technology, 441-448, 2015.

[2] A. Marphatia, A. Muhnot, T. Sachdeva, E. Shukla, L. Kurup, Optimization of FCFS Based Resource Provisioning 130 Journal of Internet Technology Volume 22 (2021) No.1 Algorithm for Cloud Computing, IOSR Journal of Computer Engineering, Vol. 10, No. 5, pp. 1-5, March-April, 2013.

[3] A. Hussain, M. Aleem, GoCJ: Google Cloud Jobs Dataset, Mendeley Data, v1, 2018, http://dx.doi.org/10.17632/ b7bp6xhrcd/

[4] Barclay. C, How to create a custom scheduler for Amazon ECS, Amazon Web Services, 2016 (On-Line): available: https://aws.amazon.com/blogs/compute/how-to-create-acustom-scheduler-for-amazon-ecs

[5] Buyya, R., Pandey, S., &Vecchiola, C, Cloudbus Toolkit For Market-Oriented Cloud Computing. In Cloud Computing (pp. 24-44). Springer Berlin Heidelberg, 2009.

[6] Buyya, R., Ranjan, R., &Calheiros, R. N. Modeling and Simulation of Scalable Cloud Computing Environments and the Cloudsim Toolkit: Challenges and Opportunities. In High Performance Computing & Simulation, International Conference on (pp. 1-11). IEEE , 2009.

[7] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., &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), 23-50, 2011.

[8] Choudhary, M., &Peddoju, S. K.A, Dynamic Optimization Algorithm for Task Scheduling In Cloud Environment. International Journal of Engineering Research and Applications (IJERA), 2(3), 2564-2568, 2012.

[9] E. Ilavarasan and P. Thambidurai, Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments, Journal of Computer Science, Vol. 3, No. 2, pp. 94-103, February, 2007.

[10] Ergu, D., Kou, G., Peng, Y., Shi, Y., & Shi, Y, The Analytic Hierarchy Process: Task Scheduling and Resource Allocation in Cloud Computing Environment. The Journal of Supercomputing, 64(3), 835-848. 70, 2013.

[11] Fang, Y., Wang, F., &Ge, J. A Task Scheduling Algorithm Based On Load Balancing In Cloud Computing. In Web Information Systems and Mining (pp. 271-277). Springer Berlin Heidelberg, 2010.

[12] Ghanbari, S., & Othman, M. A Priority Based Job Scheduling Algorithm In Cloud Computing. Procedia Engineering, 50, 778-785, 2012.

[13] H. Ali, I. A. Saroit, and A. M. Kotb, Grouped Tasks Scheduling Algorithm Based on QoS in Cloud Computing Network,Egyptian Informatics Journal, Vol. 18, No. 1, pp. 11-19, March, 2017.

[14] Han, H., Deyui, Q., Zheng, W., & Bin, F. A Qos Guided task Scheduling Model in cloud computing environment. In Emerging Intelligent Data and Web Technologies (EIDWT), 2013Fourth International Conference on (pp. 72-76), IEEE, 2013.

[15] Junwei,G. &Yongsheng,Y., Research of cloud computing task scheduling algorithm based on improved genetic algorithm, ICCSEE, 2013.

[16] ITC Group Admin, Cloud Computing: What are its Advantages and Disadvantage? Nov 2020,
https://itcgroup.io/blog/detail/cloud-computing-what-are-its-advantages-and-disadvantage_b300**.**

[17] Sasikala, P.Cloud computing: present status and future implications. International Journal of Cloud Computing, 1(1), 23–36, 2011.

[18] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu, An efficient k-means clustering algorithm: analysis and implementation, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 7, pp. 881-892, July, 2002.

[19] Tahani Aladwani Mecca, Saudi Arabia, Types of Task Scheduling Algorithms in Cloud Computing Environment, 2020.
http://creativecommons.org/