Analysis and Development of Efficient Task Scheduling Strategies in Heterogeneous Cloud Environment

Shailendra Singh

Department of Computer Science and Engineering
National Institute of Technology Rourkela

Analysis and Development of Efficient Task Scheduling Strategies in Heterogeneous Cloud Environment

Thesis submitted in partial fulfillment

of the requirements of the degree of

Master of Technology

in

Computer Science and Engineering
(*Specialization*: *Computer Science*)

by

Shailendra Singh

(Roll Number: 711CS1095)

based on research carried out

under the supervision of

Prof. Dr. Pabitra Mohan Khilar

May, 2016

Department of Computer Science and Engineering
National Institute of Technology Rourkela

Department of Computer Science and Engineering
National Institute of Technology Rourkela

Prof. Dr. Pabitra Mohan Khilar

May 19, 2016

Supervisor's Certificate

This is to certify that the work presented in the dissertation entitled *Analysis and Development of Efficient Task Scheduling Strategies in Heterogeneous Cloud Environment* submitted by *Shailendra Singh*, Roll Number 711CS1095, is a record of original research carried out by him under my supervision and guidance in partial fulfillment of the requirements of the degree of *Master of Technology* in *Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

_____
Dr. Pabitra Mohan Khilar

Dedication

dedicated to my beloved parents and siblings ...

Declaration of Originality

I, *Shailendra Singh*, Roll Number *711CS1095* hereby declare that this dissertation entitled *Analysis and Development of Efficient Task Scheduling Strategies in Heterogeneous Cloud Environment* presents my original work carried out as a postgraduate student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections "Reference" or "Bibliography". I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

May 19, 2016
NIT Rourkela

Shailendra Singh

Acknowledgment

I want to express my most profound feeling of appreciation to my guide Prof. Dr. Pabitra Mohan Khilar, Department of, NIT Rourkela, for his significant direction and help all through the consummation of this project. I express gratitude toward him for his capable direction and effective advice in enhancing my comprehension of this project. I state my feeling of utmost respect towards those whose names are taken in the reference section. I recognize my obligation to every one of them. Finally, I feel special to have such a companion circle who provided a wide range of assistance for effectively completing this project. —

May 19, 2016
NIT Rourkela

Shailendra Singh
Roll Number: 711CS1095

Abstract

In recent years, Cloud computing has become the integral part of information technology. Lots of research is being done from academic level to industry level. Cloud computing provides service to the users through internet and other distributed network environment on pay as you use basis and user demand basis. It provides an virtual environment of computing resources which can be utilized by cloud users and cloud applications. Cloud technologies are efficient, low cost and reliable that is why cloud computing has become very interesting and emerging computing paradigm for innovations. Scheduling in cloud systems is one of the biggest challenge. An efficient task scheduler is that which is flexible according to the changing environment of clouds and complexity of the submitted tasks. Efficient use of system and getting highest performance of the system is the primary goal of any task scheduling algorithm.

Cloud service providers always struggles with problems such as load balancing, Task completion time and wastage of resources. This thesis basically focuses on Task completion time of tasks submitted to the virtual Machines (VMs). Multiple experiments has been performed in CloudSim 3.0.3 simulation toolkit. All the experimental results have been obtained from CloudSim by using base classes and libraries provided in toolkit. Without using any single physical machine CloudSim library gives an full environment for development and research the different techniques for simulation and modelling.

Few most generic task scheduling strategies have been studied for this thesis. Based on the study a new strategy has been proposed. This new strategy is named as SCHFMC algorithm, it's description and study has been provided in chapter 4. SCHFMC algorithm helps in allocating the tasks to the virtual machines (VMs) with varying processing capacity. It has an efficient way to utilise the full processing power of machine so that system can be active and alive without any failure. This algorithm reduces the total completion time of all tasks submitted to the virtual machines. This algorithm has performed better than generic task scheduling methods. This new scheduling technique has reduces the total execution time of all task by upto 30% than generic methods.

***Keywords*: *Cloud Computing*; *Task Scheduling*; *Minimizing Task completion time*; *Virtual Machines*; *Load Balancing*; *CloudSim*; *Cloudlets*.**

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Cloud Computing

Cloud computing provides the infrastructure, platform and numerous software services, which is a pay-as-you-go model to provide consumers with a subscription-based service[1-3]. Formerly these services are known as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The importance of these services, highlighted in the recent report of Berkeley[1]: ”Cloud computing, long-held dream of computing as a tool, it is possible to change a large part of the IT industry, making the software more attractive as a potential services ”. Creating efficient and reliable Datacenters by giving them capabilities of virtual services like hardware, web applications, databases, web storages is the main aim of the cloud computing system. User from all around globe who can access internet can also use these utilities to create their own application and make them public over internet or they can create their own virtual computer system and can perform all computation which they can do in their local computer system.

Users do not have to spend anything for physical infrastructure. They only have to pay a small pre-agreed amount according to the Quality of Service requirements. It helps the organisation to maintain their primary focus on the innovation and smooth development. In cloud computing environment cloud infrastructure like hardware resources, storage, content delivery etc. or other services is provided by cloud service provider like Amazon, Google, Microsoft. A remote user who has access to these cloud providers can use available service as he needs and can make them free once no more use of those services. To meet the goal of delivering on-demand and high performance computing resources and services few the favourite research areas of cloud computing are –

- Efficient Datacenter organisation
- Virtualization
- Resource allocation
- Task scheduling

- Reliability of Quality of service (QoS)
- Security and Privacy

Hosting web services, delivering content to the other web applications and other real time data processing and applications are the few of the most benefited cloud based application. Practically all of these applications has different requirements of resources and need different kind of processing machine for execution in different kind of processing environment.

Maintaining the performance of the resource allocation and task scheduling policies in the real cloud system for application of varying configuration and service requirements, is very challenging due to -

- Clouds experiences the frequently changing pattern of demand of services and delivery of services, and size of the requested system.
- Highly competing QoS need and large variation in requirements.

This chapter has five other sections. Section 1.2 gives very short description about task scheduling in cloud environment. Section 1.3 demonstrate the motivation for further part of this thesis. Section 1.4 shows the actual problem statement for this thesis and finally Section 1.6 concludes the chapter.

1.2 Task Scheduling in Cloud Computing

Task scheduling is one the most important area of research to explore. In cloud computing point of view, scheduling problem is related to assign virtual machines (VMs) to the submitted tasks/jobs. An efficient task scheduler takes the task and assigns it to the such virtual machine (VM) which reduces the overall executing time and fully utilises machine. An optimal task scheduler in cloud system has responsibility that total system requirements reduces and overall cost of processing should also be decreased significantly.

Task scheduling in cloud computing system is divided into three phases [18] as shown in Figure 1.1.

- Getting the available resources (VMs) and refine them
- Select the most suitable resource according to heuristic approaches
- Allocate the task to this resource (VM).

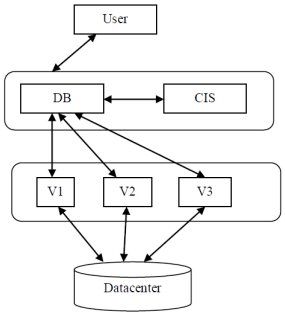

Figure 1.1: Phases of Task Scheduling

1.3 Motivation

Cloud computing has become one the most important field of computer science. Resource allocation and Task scheduling are two of the most challenging issues in any cloud system. Resource allocation consist of the two phases

- Creating virtual machine and allocating to the hosts
- Optimising the allocation so that hosts in Datacenters can be fully utilised.

The efficiency of any Task scheduling strategy depends upon the resource allocation in that cloud system. If a very small task has been submitted to the machine with large processing power than that machine will sit idle for a while. This leads to the bad utilisation of resources. Every task scheduling methodology must consider the these following protocols –

- Must improve the proper utilisation of resources (VMs) which are assigned to execute the task.
- There must be a priority in assigning certain task so that they can be executed properly and fast.
- Must increase the overall system utilisation.
- Minimizes the execution time of each task and overall completion (Make Span) of the total task submitted to cloud system.
- There must be very less faults during processing and self re-allocation of resources in case of failures.

1.4 The Problem Statement

The overall problem of task scheduling is to assign a virtual machine (VM) to the submitted tasks such that overall utilisation of the system increases and task completion time decreases. This is the main aim in cloud computing environment to allocate machines to the task effectively.

This mathematical formula is given for any task scheduler[14]. A new scheduling heuristic must be developed taking parameters of this formula in consideration –

$$\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij} * T_{ij} + X_{ij} * C_{ij} + X_{ij} * S_{ij} \leq F_{min}$$

Where each of the symbol represents -
M = 1,2,3, ... n are the n virtual machines (Resources)
N = 1,2,3 ..... m are the m task which are to be assigned to VMs
F_{min} = Objective function with minimum completion time of all tasks
X_{ij} : Task i occupies the virtual machine j
T_{ij} : Execution time of task i by the virtual machine j
C_{ij} : Cost of task i during execution by the virtual machine j
S_{ij} : Resource utilisation of of machine j by task i

Every task scheduling algorithm is developed by taking one the above three parameter (T_{ij}, C_{ij} or S_{ij}) into consideration. Every scheduler has to decrease T_{ij} and C_{ij} as much as possible and have to increase S_{ij} upto greater extent.

The main problem statement for this thesis can be briefed in following points:

- Let VMlist = VM_1, VM_2, VM_3 ..... VM_{n} which is stored into a list and can be used in future during allocation.
- Let Jlist = J_1, J_2, J_3 ..... J_{m} is the list of the m jobs and stored into a list.
- We have to map Jlist $\Longrightarrow$ Vmlist in such a way that overall completion time of all the job reduces and the use of the resources reduced so that they can serve other jobs in same time with using any other resource.

1.5 Structure of thesis

This thesis has 6 chapters including this introduction chapter. Chapter 2 gives the brief introduction of cloud computing, CloudSim and related terminology. It also gives the main characteristics and issues in development of the scheduling algorithms. Chapter 3 contains the previous work and algorithms of assigning task into the virtual machines. Chapter 4 has

detailed description of proposed algorithm for new task scheduling algorithm for assigning jobs into virtual machines. This chapter also contains comparison of the proposed algorithm with the various other algorithms and explains the goodness of the algorithm. Chapter 5 covers the simulation and results. It contains graphs and tables of the results of comparison with other algorithms. Chapter 6 concludes the work and proposes some future work.

1.6 Conclusion

The overall motive of a cloud system is to reduce overall processing cost and efficient use of the resources. Scheduling techniques should be developed in such a way that there is an even distribution of the load into the virtual machines. Primary goal of task allocation is to reduce the overall completion time and proper utilisation of virtual machines. Proposed algorithm has been developed by taking these two issues into consideration. This proposed algorithm has performed better than many algorithm discussed in chapter 3.

Chapter 2

Literature Review

2.1 Cloud Computing: Overview

The clouds are changing the way computer system being used. It has been explored from business uses to the academic uses to the personal uses. Hence without cloud computing technologies our today's life would not have been like this. All services like facebook, gmail, whatsapp etc would not been possible without cloud technologies. Cloud technologies has transformed the business section tremendously. Today numerous organisation rely on cloud services from online meeting to project collaboration and content sharing. Few of the most exciting features of cloud computing are listed below.[8-9]

This chapter gives complete literature review for this thesis. Section 2.2 gives description of available features of current cloud computing environment. Section 2.3, Section 2.4 and Section 2.5 gives the brief information about standard cloud model given by NIST, America. Section 2.6, Section 2.7. Section 2.8 and Section 2.9 has proper description about Cloudsim and how actual cloud environment can be simulated in Cloudsim. Finally Section 2.10 concludes this chapter.

2.2 Features of Cloud Computing

2.2.1 Fast Implementation

Previously there used to be various phases to develop any software or any applications. It used to takes month and even years to become public and usable. But today we can create required application using built in tools, utilities ans services available in clouds and can create applications within weeks and even in hours.

2.2.2 Less Cost

There used to be various type of expenses in building even small application from development to the deployment. But in cloud computing technologies the cost is

dramatically decreased and sometimes its completely free. We may have to pay some monthly subscription fee and at any time we can use any services available at cloud provider.

2.2.3 Scalability

There was a time when if we have to increase our system requirement we had to start from scratch. For example an any instant our used databases overflowed, than we have to use new storage infrastructure and have to take care the old data by own. But in cloud technology we do not have to do anything just have to inform the provider and pay extra cost if applicable and within minutes our storage is expanded.

2.2.4 Maintenance

This is fact that in software industry maintenance cost of any software is way higher than production cost. Cloud computing technology has proved it self the perfect solution for maintenance problem. You get regular notification of your application, statistics and vulnerabilities. It also suggests the proper methods to make application up and running smoothly.

2.2.5 Accessibility

This is one of the most important feature of the cloud technology that we do not have to find any particular place with some resources, simply if we have access to the cloud service provided we can access anything about application.

2.2.6 Security and Privacy

Most business organisation have great concern about security and privacy of their application and its data. But their has been developed many techniques to secure the data. Even today end-to-end encryption techniques has been developed which means that even cloud service provider does not have any information about data store.

The US National Institute of Standards and Technologies (NIST) has developed an model for cloud computing which contains 4 deployment models, 5 characteristics and 3 service models as shown in Figure 2.1 [4].

2.3 NIST Cloud Model: Essential Characteristics

NIST cloud model has proposed five characteristics of any cloud system[4].

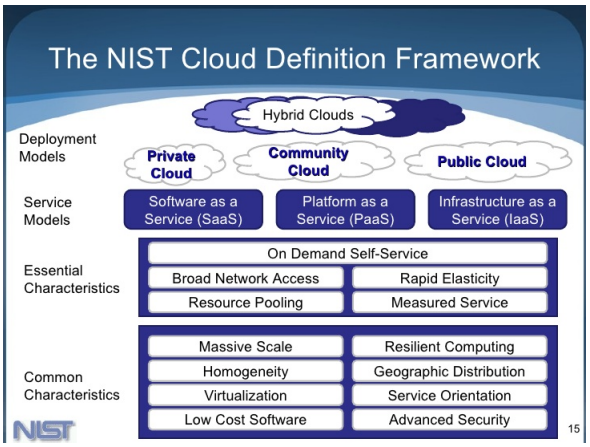

Figure 2.1: NIST Standard Cloud Computing Model

2.3.1 On-demand self service

User can request for any service available and which comes under service level agreement between user and cloud provider. And cloud must be able to deliver requested services without any human intervention.

2.3.2 Broad Network Access

Cloud services must be available through range of available network for example it must be access through mobile phone with same convenience as it is accessed via laptop or PCs.

2.3.3 Resource Pooling

The resources available on clouds are utilised in such a way that they can serve multiple users at a time. This process work on multi-tenant model. Different resources over different locations are used and re-used according to demand at that time. Customer has no knowledge about that where physical systems are. CPU, bandwidth, virtual machines ans storage are few resources.

2.3.4 Rapid Elasticity

In current information age when million of users are using services at same time there is possibility that user needs some large amount of service within limited amount of time. Cloud provider must be able to deliver the unlimited amount of service to unlimited number of user at the same time.

2.3.5 Measured Service

Cloud system should have capability to automate the process of resource optimization and its uses. There must be a transparency between cloud provider and the customer over resource utilization.

2.4 NIST Cloud Model: Service Models

NIST cloud model has proposed three service of any cloud system[4].

2.4.1 Cloud Software as a Service (SaaS)

Cloud provider has various applications running on servers. Cloud user can use these application via their web browsers of mobile or laptops. Customer does not have to care about background function or other infrastructure like operating system, memory and servers. User can start using these services simply authenticating the service. For example: Gmail service. There is no loss of information or data because everything is stored and processed over clouds if local device (mobile or laptops) stops working or lost than do not have to worry, simply login again with other device and start working from where left last time. Services can be extensively scaled as consumer needs.

2.4.2 Cloud Platform as a Service (PaaS)

This service provides all in one package to develop and deploy the cloud based (Web based) application. By taking simple example of a ASP.Net based website with MySQL Server as database. Cloud provider like Amazon Web Services (AWS) gives all in one package like web templates in HTML and CSS, Integrated Development Environment (IDE) to write ASP code with HTML, CSS and JavaScript. And access to a database based on MySQL server with latest version available. And great thing about PaaS is that all these things are made available through web browser. We do not have to go anywhere or we do not have to install MicroSoft Visual Studio to for development. Once the development is done AWS gives cheap hosting services to host our ASP website. This is the power of PasS of Cloud Service. It reduces the complexity of the middle ware from development to deployment.

2.4.3 Cloud Infrastructure as a Service (IaaS)

This service of cloud computing basically has been developed taking big organisation into consideration. This service provide the organisation the huge processing resources like Data Centres, large network storage and servers as pay per use basis. It reduces huge cost by

zero investment in hardware infrastructure. The whole infrastructure scales as demanded by customer to help the varying workload of cloud system

2.5 NIST Cloud Model: Deployment Model

Deployment model considers that when and where these cloud will be available ans who will be the owner of the cloud. According to these parameters cloud has been derided into four categories. NIST cloud model has proposed four service of any cloud system[4].

2.5.1 Public Cloud

Public clouds are the proprietary of companies which has frequent access to the public network and processing resources in low cost. User do not have to worry about supporting resources or they do not have to buy any hardware and software, they are managed and owned by cloud providers. SaaS application from CRM to data analytics systems. It also has powerful PaaS for web based application development and deployment model. It also provided elastic and scalable IaaS services.

2.5.2 Private Cloud

The sole purpose of private cloud is to serve single organisation. It may be managed by same organisation or by some third party provider. These clouds can be hosted either inside the same organisation or at some external sources. There is an automated interface which controls the services. It allows organisation staff ro allocate and re-allocate the resources. It has special security and administrative protocols designed for special requirement of the company.

2.5.3 Hybrid Cloud

A hybrid cloud utilises the foundation of the private cloud attached with planned connection and takes advantage of public cloud services. The problem is that private clouds of any organisation can not be survive alone mean isolated with the organisation's resources. Hence most of the organisations with private clouds takes advantage of hybrid clouds to manage the loads across the public and private clouds and the datacenters. These clouds helps organisations to keep sensitive data in private clouds. It enables to use Public clouds as well as private cloud from same environment and resources. It also has feature of portability.

2.5.4 Community Cloud

This cloud infrastructure is allocated for use by particular community of users from some group or organisation who has shared requirements. It might be governed by same group of

community or some third party.

2.6 Cloud Computing with CloudSim

Every cloud service provider has responsibility to keep on changing their allocation, scheduling and other security related issues at certain interval of time. The developer has to develop the algorithms and application hence to test them they need a fully functional cloud computing environment. This is not practical to all level of development. It is not realistic to carry out regular experiments in repeatable and scalable environments using real world cloud systems. Hence a simulation toolkit like CloudSim can be highly useful for performing real cloud experiments into the simulation tool. CloudSim is a library written in Java language. The Cloudsim toolkit facilitates the system level and behaviour modelling of the cloud computing elements like VMs, Datacenters, hosts, resource and task allocation policies. It implements most common policies which can be further extended as developer need. Current version 3.0.3 of CloudSim supports the simulation modelling of the clouds environment having single and federation of clouds[5-6].

2.7 Layered Design of CloudSim

CloudSim toolkit is developed by taking four layers into consideration. SaaS, PaaS and IaaS and End Users are directly related to each of the layer of Cloudsim. It is shown in Figure 2.2. Each level represents the different service offering of the real cloud systems.

2.7.1 User Level Middleware (SaaS)

This layer consist of framework and web interfaces to help programmers and developers for developing and creating rich and effective application and interfaces for application running on the upper level. This layer supports other modules like programming, libraries which helps in distributed programming to create application development and deployment.

2.7.2 Core Middle ware (PaaS)

Services of platform level are implemented by this layer which helps in running and managing the user level applications. The services provided by this layer is accessed by both layer upper layer i.e. SaaS and lower layer i.e. IaaS. Real cloud example of this layer is Amazon's EC2 and Google's App Engine. Workload balance, monitoring the running services are the few of the functions of this layer. For example Amazon provides Cloudwatch (A monitoring service) for EC2 developers and Microsoft Azure's service provides the .NET service bus for message passing mechanism.

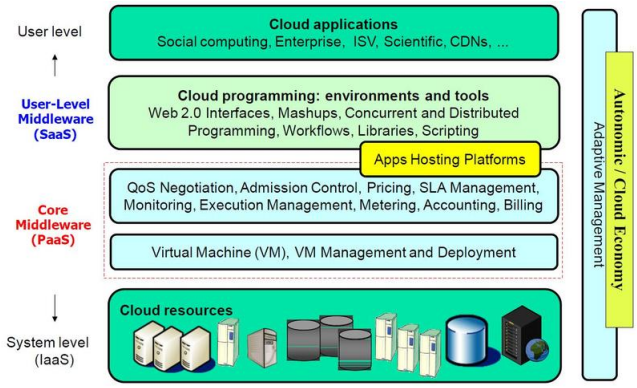

Figure 2.2: Layered Design of CloudSim

2.7.3 System Level (IaaS)

This can be called as physical resources layer. This layers helps in creating hosts and machines which helps in creating the data centres which create the foundation of cloud computing. All other layers are totally dependent upon this layer. PaaS layer uses this layer to create different policies for running the applications. Many storage and application servers are powered by this layer because this layer has massive physical resources and they are managed by higher level virtualization.

2.7.4 Cloud Application

This is upper most layer and directly communicates with User level middle layer (SaaS). These applications are directly used by end users. May be these application provided on pay per basis or some monthly/annually subscription basis. For example Amazon gives an web based platform where user can sign up and subscribe the services and start using services directly from the browser. Similarly other cloud provider like Salesforce some CRM softwares, Social networks and some content delivery networks.

2.8 Implementing Clouds in CloudSim 3.0.3

Cloudsim is a java library which provides many mini libraries, classes and abstract classes which can be extended as we need. The overall class design is shown in below Figure 2.3. Few of the frequently used classes during experiments for this thesis are explained below.

- Cloudlet : This class is most frequently used in this project. A cloudlet in CloudSim is equivalent to a task/job in real cloud system. It has many elements like file size, length, output size etc. But most commonly used element is length. When we create object of this class we have to supply length. Length of Cloudlet actually represent the size of the task. It is represented in Million Instructions (MI).

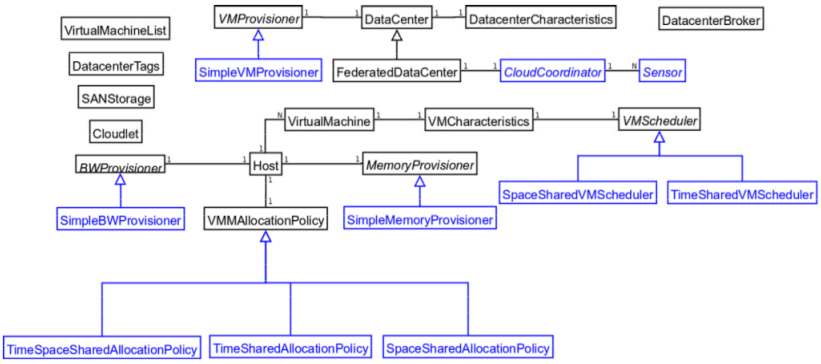

Figure 2.3: Class Diagram of CloudSim 3.0.3

- Vm: This class is also most frequently used class. A vm object represent the real virtual machine in clouds. It has components like ram, size, bandwidth and mips out of these mips is frequently used. Mips stands for million instruction per second we can directly relate it with MI of cloudlet.
- CloudletScheduler: This is an abstract class which can be extended for creating our own scheduling policies for scheduling Cloudlets into VMs. CloudletSchedulerSpaceShared and CloudletSchedulerTimeShared are the two type of scheduling policies has been offered in this class.
- Host: This class models a host. We can create any number of objects of this class each one will represent an actual physical machine with its own processing power, storage, bandwidth, storage cost etc.
- Datacenter: This class is provided to model core infrastructure. This class works with host class to create the fully function datacenters. Hosts can be heterogeneous or homogeneous according to their configuration of the MIPS, Storage and RAM etc.
- DatacenterBroker: This class work as mediator between the SaaS and the Cloud service provider. It finds most suitable clouds and ask for efficiently used resource allocation policies which can meet application requirements.

2.9 Task Scheduling Strategy and Simulation in CloudSim

The main aim of this thesis to analyse the different existing task scheduling strategies and develop new technique. All algorithms discussed in next chapter of this thesis has been implemented in this CloudSim toolkit. Most of experiment carried out for this thesis has utilises the classes discussed earlier. CloudSim made quite easy to carry out such an

experiment for variety of inputs. Multiple experiment has been carried out with changing the processing power of VMs and length of Cloudlets.[8][10-11]

2.10 Conclusion

This chapter has covered most of the literature, tools and libraries used for this project. This helps in understanding the upcoming algorithm and techniques which can be easily implemented in CloudSim. Moreover all experimental results has been generated from CloudSIm.

Chapter 3

Implementation of Generic Task Scheduling Strategies in Clouds

3.1 Introduction

Task scheduling is the method to map virtual machines to the tasks (Cloudlets). We can also say that scheduler has to choose most suitable VM for a particular cloudlet so that overall execution time for all the Cloudlets decreased and all the virtual machines do not sit idle during execution of the Cloudlets. Hence every task scheduling technique must consider the execution time of cloudlets and utilisation of VMs. The performance of the datacenters is dependent upon these assignment of VMs to the Cloudlets. Task scheduling is the NP-Complete problem. Hence task scheduling has high importance in cloud computing. There has been lot of research and many algorithms and techniques has been developed in task scheduling field[7-11].

This chapter contains five other section than this. Section 3.2 gives brief introduction about two types of scheduling used in clouds. Section 3.3 contains input data for upcoming implementation of various algorithm later in this chapter. Section 3.4 gives the system configuration of the cloud environment for simulation in Cloudsim. Section 3.5 has detailed description and their implementation of various algorithms. Finally section 3.6 concludes the chapter.

3.2 Static and Dynamic Scheduling

In cloud computing environment variety of the task scheduling algorithms has been developed. Most of them can be categorised into two types Static and Dynamic schedulers. Static schedulers has all information about task and available machines where task has to be executed. Hence scheduling is done after processing the pre-fetched information. While Dynamic schedulers do not have any information until they starts the execution of the task[15]. They can not do any pre-processing. All logic works on run time while tasks has been started executing. Run time overhead of the static schedulers is less than the dynamic

schedulers because of pre-processing phase of static schedulers. All of the studied strategies for this thesis are static scheduling techniques because all the information of the cloudlets (Task) and VMs (Resource) is already provided before starting execution phase of the cloud information service (CIS).

This thesis has explanation of few of the most common scheduling techniques, their algorithms and their experimental results on CloudSim 3.0.3. These algorithms has been implemented in CloudSim. Input for each implementation has been taken same for better understanding and comparison with each other -

3.3 System Configuration

For all algorithm which are implemented in CloudSim has following system configuration.

Two Datacenters with 2 host on each Datacenters. One of the host has quad-core processor i.e. it has four processing elements (Pe) and each Pe has processing power of 25000 MIPS. While second host is dual-core mean it has two Pes each with processing power of 25000 MIPS. Each host has RAM of 8 GB. Other parameter has been taken as default as provided into CloudSim because they are irrelevant to these experiments. Datacenter broker starts from first Datacenter for creating virtual machines. If it does not enough processing units in first Datacenter (DC) it requests to second DC and creates the virtual machine of requested configuration. At last if both DCs can not provide enough processing units to create VM of desired configuration that VM will not be created.

As creation of VM is the part of Resource Allocation problem of cloud computing, hence All experiment rely on default resource allocation mechanism provided in CloudSim, Also to avoid any case where some requested VM is not created because not enough processing units available at host level (DC). Hence enough MIPS and multi core processors has been created for performing experiments.

In next part of thesis there are few of the task scheduling strategies has been studied and few of them has been experimented fully on CloudSim. Their algorithms and experimental results are shown with their explanation in further part of this thesis. Inputs for all the experiments is given in next part of this thesis.

3.4 Input for Algorithms and their Implementations

Each experiment has been implemented in CloudSim toolkit 3.0.3. Their system configuration is same as given in section 3.2 of this thesis. Every algorithm is implemented with one broker which requests both DCs to create 6 VMs on available hosts. Their is one user which creates 20 Cloudlets (Tasks) to be executed by those 6 VMs. The full configuration of VMs and Cloudlets has been shown in following table.

Table 3.1: VMs configuration used in experiments

VM Id	MIPS	RAM(MB)	No of Pe
0	1000	512	1
1	600	512	1
2	300	512	1
3	800	512	1
4	1200	512	1
5	400	512	1

Table 3.2: Cloudlets configuration used in experiments

Cloudlet Id	MI	No of Pe
0	8000	1
1	6000	1
2	4000	1
3	6500	1
4	1200	1
5	3300	1
6	2500	1
7	4400	1
8	7700	1
9	3900	1
10	5400	1
11	1900	1
12	2300	1
13	7100	1
14	5100	1
15	4300	1
16	3500	1
17	2600	1
18	6100	1
19	8500	1

Following are the task scheduling strategies has been studied before writing this thesis. Few of them are most common techniques. Few of them are implemented in CloudSim and experimented for the range of inputs apart from the input given in this thesis.

- First Come First Serve (FCFS)
- Round Robin Scheduling (RR)
- Generalized Priority Scheduling (GPS)
- Min-Min Scheduling
- Max-Min Scheduling
- Shortest Cloudlets to Fastest Processor (SCFP)
- Longest Cloudlets to Fastest Processor (LCFP)
- Proposed Algorithm - SCHFMC

3.5 Existing Common Algorithms Implementation

3.5.1 First Come First Serve (FCFS)

This is most commonly used task scheduling strategy. As soon as any cloudlet is received in clouds it is sent to next available VMs. It does not worry about processing speed of machine or size of the task. Neither it care about execution time of the task. First task received is bind to the first machine, second task to the second machine until all machines are occupied. In such case upcoming task are schedules from first machine again. One previous task is completed by that machine this new task will be executed.
For example: 6 tasks (J_1,J_2, J_3,)J_4,J_5,J_6 will be scheduled to 3 machine (VM_1,VM_2,VM_3) in FCFS in following way -
($J_1 \Longrightarrow VM_1$) , ($J_2 \Longrightarrow VM_2$) , ($J_3 \Longrightarrow VM_3$) , ($J_4 \Longrightarrow VM_1$) , ($J_5 \Longrightarrow VM_2$) , ($J_6 \Longrightarrow VM_3$)

Results of FCFS: For the given input of Cloudlets and VMs as given in Table 3.1 and Table 3.2. The final results of allocation of VM to the Cloudlets and execution time has been given in given in the final comparison Table 3.3. The graph for the results of the FCFS algorithm has been shown in Figure 3.1.

Disadvantage of this technique is that small task which are stored after long task in the scheduled queue of a VM have to wait till all task before them in queue has been executed. In above example Task J_4 will be executed only when Task J_1 has been executed completely. So if size of task J_4 is significantly less that task J_1 still it has to wait for long time till task J_1 is completed executing.

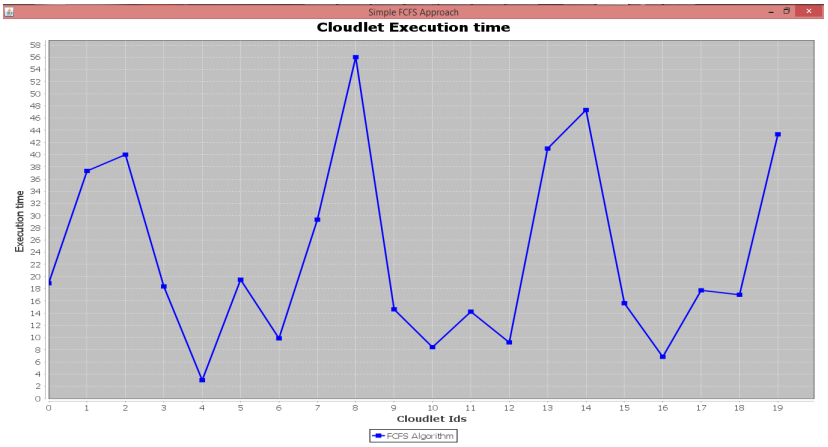

Figure 3.1: Graphical representation of FCFS results

3.5.2 Round Robin Scheduling (RR)

The main disadvantage of FCFS scheduling was that in many cases small task has to wait for long task to complete first if they are in last of the waiting queue. Round Robin scheduling technique overcome this problem by using a small concept of giving fair chance to all tasks in scheduled queue of a particular VM to be executed for equal amount of time. By using this concept the finish time of small task at the end of the queue reduces significantly. Unlike the FCFS a task is not executed completely while it is pre-empted after certain amount of time (time slice) and next task in that queue is executed for the same time (time slice). This process continues until all the tasks in the queue has been executed.

For example: 9 tasks (J_1, J_2 , J_3, ..., J_9) has to be scheduled into 3 VMs (VM_1,VM_2 and VM_3) than waiting queue for each of the VM is -

Queue for $VM_1 \Longrightarrow J_1$—-J_4—-J_7

Queue for $VM_2 \Longrightarrow J_2$—-J_5—-J_8

Queue for $VM_3 \Longrightarrow J_3$—-J_6—-J_9

Let's suppose Size of Tasks in Queue for VM_1. (J_1, 1000) (J_4, 100) (J_7, 10) and execution time is 1 unit for each size unit of tasks. Hence in simple FCFS approach the finish time of J_1 will be 1000 unit , for J_4 will 1100 unit and for J_7 finish time will be 1110 time unit. As we can see that J_7 will only take 10 unit of time. It could have been finished execution at 10 unit time after start of VM_1.

In round robin technique lets suppose time slice is 1 unit. Hence after execution of 1 size of task execution is pre-empted and next task in queue starts executing. Hence using this mechanism J_7 will finish after 30 unit time (10*3), J_4 will finished by 210 (30 + 90 * 2) time unit. And J_1 will be finished by 1110 (900 + 210) time unit.

As from comparison from both algorithm we can see that total execution time of all the task is same. But the finish time of the tasks in queues decreases significantly. Hence Round Robin scheduler has advantage if we have to take care of small job which some how

submitted later than bigger jobs.

3.5.3 Generalized Priority Scheduling

This is not an individual algorithms. This is an approach which cover many aspect of scheduling and helps developing new strategies. Every Cloudlet and VM has many characteristics. User has many requirements. In cloud environment every aspect of task and machine is important either it is processing time or resource used or which task finish first or how much memory task is taking to execute. Hence it is user's call what is his priority for execution. There are many characteristic comes into picture. Some of them are –

- MET: Minimum execution time of the task in VMs
- MCT: Minimum Completion/Finish time of tasks in VMs
- Memory: Less memory used by task for execution
- CPU Utilisation: Which task utilizes CPU more
- Overall Execution Time: Minimize total execution time of all submitted task.

There may be other parameters to prioritise the task to be executed. For example Round Robin technique priority the MCT of the task that is why concept of pre-emption and time slicing came into picture. Upcoming algorithm of this thesis like SCFP, LCFP , Min-Min, Max-Min and proposed algorithm has given some priority to the execution time and MCT. It may be MET or overall execution time, depends upon individual algorithm and its approach.

3.5.4 Min-Min Task Scheduling

This strategy does not bother about configuration of VMs because it does an exhaustive pre-processing of task execution. This algorithm can be seen as just next approach after Round Robin. In Round Robin approach we basically consider the finish time of the task and we do not worry about small task executed first or last[25]. But min-min algorithm takes care of execution of small task in small time as well as they executed first. Hence both problem is solved using this algorithm without introducing concept of time slice

Following Min-Min Algorithms has an ETC a matrix created for execution time of completion. This matrix stores the execution time of each task (J_{i}) in each virtual machine (VM_{j}).

Min-min algorithm takes remaining task in each mapping of the tasks till all the tasks are not assigned. Hence this algorithm will execute small task first followed by larger tasks. There is high possibility that there may be an load imbalance if majority of the tasks are larger. The time complexity of this algorithm reaches to O(J^2m) where j is number of tasks and m is number of VMs. This algorithm give great results if ETC matrix has very less bigger execution time value or in direct words if there less larger jobs to be executed.

Algorithm 1

```
1: procedure MIN-MIN ALGORITHM
2:     cloudletList(CL) ⟸ CL1, CL2, CL3, ..... ,CLn
3:     vmList(VM) ⟸ VM1,VM2,VM3, ..... VMm
4:     while notEmpty(cloudletList) do
5:         for each Task(CLi) in cloudletList(CL) do
6:             for each machine(VMj) in vmList(VM) do
7:                 Execij ⟸ Calclulate execution time of task CLi in machine VMj
8:                 ETCij ⟸ Execij
9:             end for
10:        end for
11:        Find Minimum value(ETCij) in ETC matrix
12:        Assign CLi ⟸ VMj
13:        Delete CLi From cloudletList(CL)
14:    end while
15: end procedure
```

3.5.5 Max-Min Task Scheduling

This algorithm is same as Min-min algorithm except one stage of the algorithm. This algorithms gives priority to the execution time of the task as well as bigger task executed first. It also does not takes care about any system configuration of virtual machines. It also does the same pre-processing as min-min does that is creation of the ETC matrix where execution time of each task on each machine is calculate. This matrix is refined after every mapping step of the algorithm[23-24]. ETC matrix is same as explained in min-min algorithm. We just select maximum completion time on each mapping of task because this algorithm gives priority to the larger jobs first, small jobs are executed later than bigger jobs.

Max-min algorithm takes remaining task in each mapping of the tasks till all the tasks are not assigned. Hence this algorithm will execute larger task first followed by smaller tasks. There is high possibility that there may be an load imbalance if majority of the tasks are smaller. The time complexity of this algorithm reaches to O(J^2m) where j is number of tasks and m is number of VMs. This algorithm give great results if ETC matrix has very less smaller execution time value or in direct words if there less smaller jobs to be executed.

Hence min-min and max-min algorithm are almost same and should be used according the input jobs. If priority is to given to smaller jobs than Min-min strategy should be used while if larger job has to executed first than Max-min algorithm should be used.

3.5.6 Largest Cloudlets to Fastest Processor (LCFP)

All algorithms which has been discussed till now have not taken care much about information related to machine configuration. There has been series of experiment has been carried out and found that if we give priority to the processing power of the VMs and than after we

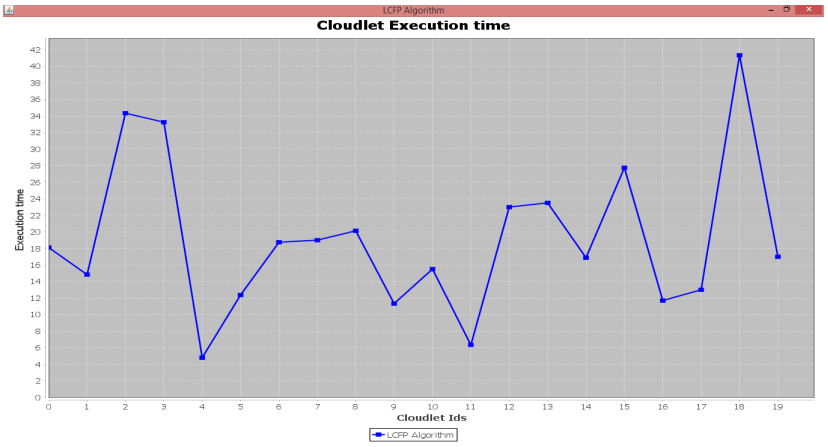

Figure 3.2: Graphical representation of LCFP results

schedule the task to them in FCFS manner than this simple approach give much better results than normal FCFS algorithm. Also in series of experiment it also has been observed that if we use faster machines first than overall execution time decreases and utilisation of machine is increases. Hence two basic approaches can be tested. One them is LCFP which targets tasks with large size should be scheduled to faster machines. The result of this algorithms are better than normal algorithm.
Step for this approach are -

- Sort in Descending cloudletList according to the size (MI) of cloudlet.
- Sort in Descending vmList according to the processing power (MIPS) of VMs
- Allocate sorted cloudlets to the sorted VMs in FCFS way.

3.5.7 Smaller Cloudlets to Fastest Processor (SCFP)

In other experiment when we schedule smaller tasks into faster machines than every time it gave better results than LCFP and much better results than FCFS algorithm. Following are the three basic step for implementation of this algorithm.

- Sort in Ascending cloudletList according to the size (MI) of cloudlet.
- Sort in Descending vmList according to the processing power (MIPS) of VMs
- Allocate sorted cloudlets to the sorted VMs in FCFS way.

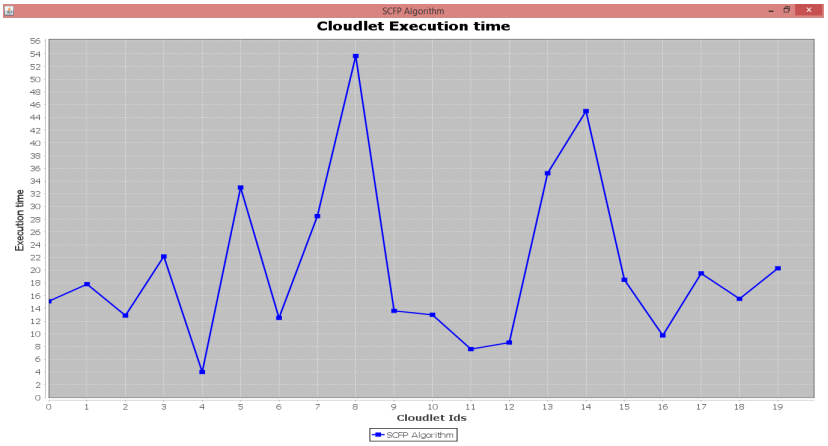

Figure 3.3: Graphical representation of SCFP results

Table 3.3: Comparison of FCFS, LCFP, FCFP with input vmList and cloudletList

Cloudlet Id	MI	FCFS	FCFS	LCFP	LCFP	FCFP	LCFP
0	8000	0	18.92	0	18.1	4	15.12
1	6000	1	37.33	4	14.83	0	17.8
2	4000	2	40	2	34.33	3	12.87
3	6500	3	18.37	5	33.24	1	22.16
4	1200	4	3	0	4.8	4	4
5	3300	5	19.5	3	12.37	2	32.99
6	2500	0	9.86	5	18.74	1	12.5
7	4400	1	29.33	1	19	5	28.49
8	7700	2	56	3	20.12	2	53.66
9	3900	3	14.62	4	11.33	0	13.61
10	5400	4	8.42	0	15.5	4	12.98
11	1900	5	14.24	4	6.33	0	7.6
12	2300	0	9.2	2	22.99	3	8.62
13	7100	1	41	1	23.5	5	35.24
14	5100	2	47.33	3	16.87	2	45
15	4300	3	15.62	5	27.74	1	18.5
16	3500	4	6.83	0	11.7	4	9.75
17	2600	5	17.75	1	13	5	19.49
18	6100	0	17.02	2	41.33	3	15.5
19	8500	1	43.33	4	16.98	0	20.3
Total	94,300		467.67		382.82		406.18

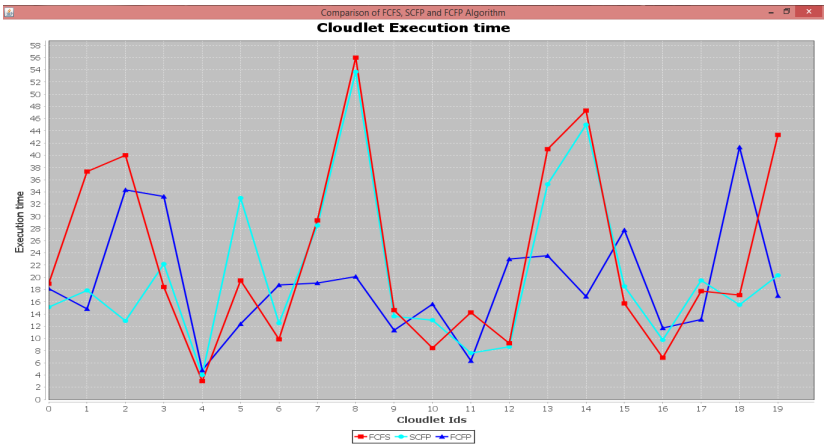

Figure 3.4: Graphical representation of Comparison of FCFS,SCFP,LCFP results

3.6 Conclusion

Previously discussed generic task scheduling methods gives priority to the task and its completion time. Problem of small task being executed at last in FCFS is solved by Round Robin method by introducing the time slice method. Min-min and Max-min methods takes of small job and bigger jobs schedules first respectively. Only LCFP and SCFP methods only considers VMs capacity. So task scheduling method has been introduced in next chapter which not only takes care about smaller task schedules first while to distribute the load to each VM properly.

Chapter 4

Proposed Work

4.1 Introduction

All previously discussed task scheduling strategies are being used in current cloud computing system according to requirement and priority of the task defined by the end user requesting cloud services. All of the algorithms discussed in last chapter only consider one aspect of QoS at a time. Either it reduces execution time of tasks or it improves the utilisation and load balance of the VMs. A new task scheduling strategy has been proposed which decreases total execution time of all tasks submitted to VMs and at the same time utilisation of VMs is also improved. The VM utilisation is studied in terms of makespan which is the total time VMs has executed task till all tasks in all VMs get executed. Hence makespan of the all VMs is different and becomes one of the most important parameter to improve to overall Quality of Service of any Cloud Provider. In other words better makespan of machines represents that VMs did not sit idle during execution process of tasks. Makespan is also defined and explained in next part of this chapter.

This chapter is divided into 6 sections excluding this introduction section. Section 4.2 contains the different notations and parameters to understand the proposed algorithm(SCHFMC). Section 4.3 contains actual algorithm. Section 4.4 briefly explains the SCHFMC algorithm and its working. Section 4.5 contains information about implementation in Cloudsim 3.0.3. Section 4.6 has detailed comparative study of four algorithm SCHFMC,FCFS,SCFP,LCFP with simulation results in Cloudsim and at the end Section 4.7 Concludes the chapter.

4.2 Notations and Performance Parameters

For better understanding of proposed algorithm following are the different term and their explanation used in the proposed work:

4.2.1 Task Execution Time (ET)

Task execution time (ET) is the execution time of any task by a VM which has processed it. It is also the ratio of the size of the task by the processing power of the VM. As this experiment is carried out in CloudSim. If task(Cloudlet) i is executed by VM j than it can be defined as below
ET_i = Task Execution time of the $Task_i$ in VM_j = MI of the $Cloudlet_i$ / MIPS of the VM_j

4.2.2 Task Start Time (ST)

As there are many tasks are scheduled at time to the VMs. So tasks are stored in Queue and one by one tasks are executed hence start time for the each task varies. The first task which began executing has Starting Time (ST) as zero. Next task in queue has Start time is same as finished time of previously executed task in the queue.

4.2.3 Task Finish Time (FT)

As VMs execute task one at a time. Hence tasks which are in queue has different finished time. Which task executed first has less FT than which is execute later.
FT = ST + TE

4.2.4 Fractional Task Length (FTL)

FTL of any task in clouds represents the contribution to the total size of all task submitted for processing at given time for the execution by the virtual machines. Suppose there are n task (Cloudlet) having length of L_1, L_2 , L_3 ..... L_n.

Hence, FTL_i = Fractional Task Length of Task i = L_i * 100 / (L_1 + L_2 + L_3 ..... + L_n)
Where L_i represent Length or size of task i.

4.2.5 Fractional Machine Capacity (FMC)

FMC of a Virtual Machine represents the contribution of its capacity to the the total processing capacity of all the VMs combined. Suppose there are m VMs with processing power of P_1, P_2, P_3 ..... P_m respectively. Then
FMC_j = Fractional Machine Capacity of Virtual Machine j = P_j * 100 / (P_1 + P_2+P_3 + ...+ P_m) Where P_j represent the processing power of Virtual Machine j.

4.3 Algorithm

Algorithm 2

procedure PROPOSED ALGORITHM - SCHFMC
    cloudletList(CL) $\Longleftarrow$ $CL_1, CL_2, CL_3, \ldots\ldots, CL_n$
    CLtotal $\Longleftarrow$ $CL_1 + CL_2 + CL_3 + \ldots\ldots + CL_n$
    vmList(VM) $\Longleftarrow$ $VM_1, VM_2, VM_3, \ldots\ldots VM_m$
    VMtotal $\Longleftarrow$ $VM_1 + VM_2 + VM_3 + \ldots\ldots + VM_m$
    FTLlist = CreateFTLlist(CL,CLtotal)
    FMClist = CreateFMClist(VM,VMtotal)
    SortedFTLlist = Sort FTLlist in Ascending order of FTL value of each Cloudlet
    if $(n == 1 \quad AND \quad m == 1)$ **then**
        $CL_1 \leftarrow VM_1$
    else
        for *each Cloudlet(CL_i) in SortedFTLlist* **do**
            Find the VM_j corresponding to maximum FMC value from FMClist
            Assign the Cloudlet(CL_i) $\Longleftarrow$ VM_j
            Update FMClist (FMC_j = FMC_j - FTL_i)
            Where FTL_i is FTL value of Cloudlet CL_i
        end for
    end if

    function CREATEFTLLIST($CL, CLtotal$)
        CLlist $\Longleftarrow$ *A New List*
        for *each Cloudlet in CL* **do**
            $FTLlist_i \Longleftarrow CL_i$ / CLtotal
        end for
        return *FTLlist*
    end function

    function CREATEFMCLIST($VM, VMtotal$)
        VMlist $\Longleftarrow$ *A New List*
        for *each Vm in VM* **do**
            $FMClist_i \Longleftarrow VM_i$ / VMtotal
        end for
        return *FMClist*
    end function
    End
end procedure

4.4 Breif Explantion

SCHFMC algorithm is based on the concept, Execute as many tasks as possible in same Machine. This scheduling technique finds the FTL of each task and FMC of each VMs. Here

FTL_{i} = (MI of the Cloudlet i * 100) / Sum of MI of all the Cloudlets

FMC_{j} = (MIPS of virtual machine j * 100) / Sum of MIPS of all VMs

First of all, Cloudlets are sorted in ascending order so that smaller task can be scheduled first. Than cloudlets are scheduld in FCFS manner with pre-processing steps. First task from the sorted cloudlet list is assigned to th VM with the highest FMC. Next the FTL of this task is subtracted from the FMC of that VM. FMC list is updated again and VM with highest FMC value id found for second task in sorted cloudlet list. These steps keep on repeating until all the task is assigned to some VMs. As it is clear that in each task mapping stage the FMC value of the VM which is involved in that step is decreased and the FMC of other VMs is not touched. By this approach load of each machine is equally distributed according to the submitted tasks.

4.5 Implementation and Results

00 Proposed algorithm(SCHFMC) has been implemented in Cloudsim 3.0.3 with same inputs as in Table 3.1 and Table 3.2. Hence according to algorithm FTL list has been created in Table 4.1 while the sorted list has been show in Table 4.2. FMC list has been created in Table 4.3 and Table 4.4 represents the sorted version of Table 4.3 according to the FMC value of each VM.

Table 4.6 represents the final allocation table for the given input to the proposed algorithm. The graphical form of results has been shown in Table 4.5. From the results Table 3.3 of FCFS,FCFP and LCFP it is clear that proposed algorithm(SCHFMC) has performed much better than all the algorithms.

Proposed algorithm(SCHFMC) get 9% improvement over FCFP algorithm, 15% improvement over LCFP algorithm and almost 26% improvement over FCFS results because of the fact that proposed algorithm(SCHFMC) utilises the FMC value of each VM before assigning each task to VM. The load of all the task has been equally distributed to the each VM. Each VM gets a cloudlet until its FMC value does not become lower than other VMs.

Table 4.1: Cloudlet/Task list with FTL value

Cloudlet Id	MI	FTL
0	8000	8.48
1	6000	6.36
2	4000	4.24
3	6500	6.89
4	1200	1.21
5	3300	3.51
6	2500	2.65
7	4400	4.67
8	7700	8.17
9	3900	4.14
10	5400	5.73
11	1900	2.03
12	2300	2.44
13	7100	7.53
14	5100	5.41
15	4300	4.56
16	3500	3.72
17	2600	2.76
18	6100	6.47
19	8500	9.03
Total	94,300	100%

Table 4.2: Cloudlet/Task list in sorted order according to FTL value

Cloudlet Id	MI	FTL
19	8500	9.03
0	8000	8.48
8	7700	8.17
13	7100	7.53
3	6500	6.89
18	6100	6.47
1	6000	6.36
10	5400	5.73
14	5100	5.41
7	4400	4.67
15	4300	4.56
2	4000	4.24
9	3900	4.14
16	3500	3.72
5	3300	3.51
17	2600	2.76
6	2500	2.65
12	2300	2.44
11	1900	2.03
4	1200	1.21
Total	94,300	100%

Table 4.3: VM list with their FMC value

VM Id	MIPS	FMC (%)
0	1000	23.26
1	600	13.95
2	300	6.97
3	800	18.61
4	1200	27.91
5	400	9.30
Total	4300	100%

Table 4.4: VM list in sorted order according to FMC value

VM Id	MIPS	FMC (%)
4	1200	27.91
0	1000	23.26
3	800	18.61
1	600	13.95
5	400	9.30
2	300	6.97
Total	4300	100

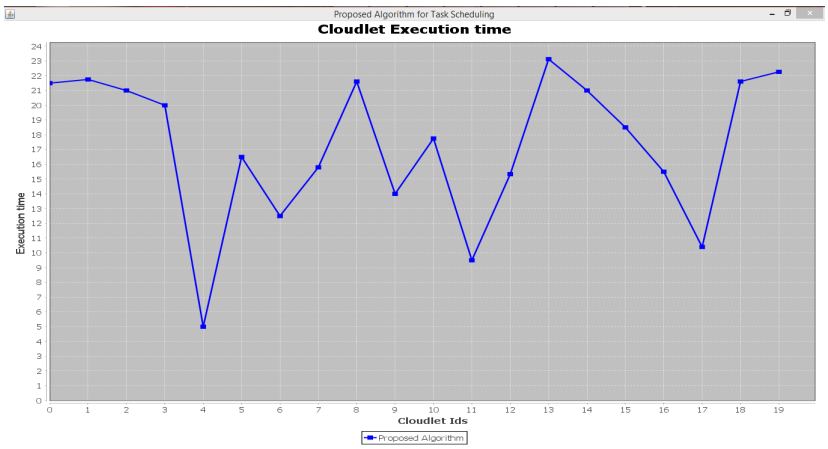

Table 4.5: Graphical representation of Proposed algorithm results

Table 4.6: Final allocation table with assigned VM and Execution Time (TE)

Cloudlet Id	MI	FTL	VM id	TE
19	8500	9.03	4	22.26
0	8000	8.48	0	21.5
8	7700	8.17	4	21.61
13	7100	7.53	3	23.12
3	6500	6.89	0	20
18	6100	6.47	1	21.61
1	6000	6.36	3	21.75
10	5400	5.73	4	17.75
14	5100	5.41	5	21
7	4400	4.67	0	15.8
15	4300	4.56	1	18.5
2	4000	4.24	2	21
9	3900	4.14	4	14
16	3500	3.72	3	15.5
5	3300	3.51	5	16.5
17	2600	2.76	0	10.4
6	2500	2.65	1	12.5
12	2300	2.44	2	15.33
11	1900	2.03	3	9.5
4	1200	1.21	4	5
Total	94,300	100%		344.63

4.6 Comparative Study: FCFS,LCFP,SCFP,SCHFMC

In this section, Proposed algorithm(SCHFMC) along with FCFS,LCFP and FCFP is compared for same input data. In previous section all algorithms has been studied and their simulation results also has been demonstrated separately but here results are compared with each other. There are three experiments have been carried out with different input data for algorithms. Experiment-1 is simulated for 5 VMs and 20 Cloudlets, Experiment-2 is simulated for 10 VMs and 50 Cloudlets and Experiment-3 is simulated for 20 VMs and 200 Cloudlets. For Experiment-1 and Experiment-2 final results are shown as allocation table with execution time and graphical representation of results. While for Experiment-3 only graphical representation is given.

4.6.1 Experiment 1

This experiment is carried out for the 5 virtual machines and 20 cloudlets. All four algorithms have been implemented for the same set of inputs of VMs and Cloudlets. The configuration of VMs and Cloudlets has been give in Table 4.7 and Table 4.8 respectively. Final results with execution time of each task from algorithm has been given in Table 4.9. The graph of final results is shown in Figure 4.1.

With only 20 cloudlets proposed algorithm(SCHFMC) performed better than previous algorithms. It has improved total execution time of all the cloudlets by 15% compared to FCFS algorithm. Also it has improved time by 25% than FCFP and by 11% over LCFP algorithm.

Table 4.7: VM list with MIPS capacity for Experiment-1

VM Id	0	1	2	3	4
MIPS	1390	103	217	1279	1029

Table 4.8: Cloudlet list with MI length for Experiment-1

CLID	0	1	2	3	4	5	6	7	8	9
MI	2773	9706	6839	4087	8324	10405	11566	6130	6197	9175
CLID	10	11	12	13	14	15	16	17	18	19
MI	3098	10347	2966	9354	5229	3726	11045	6393	6748	8815

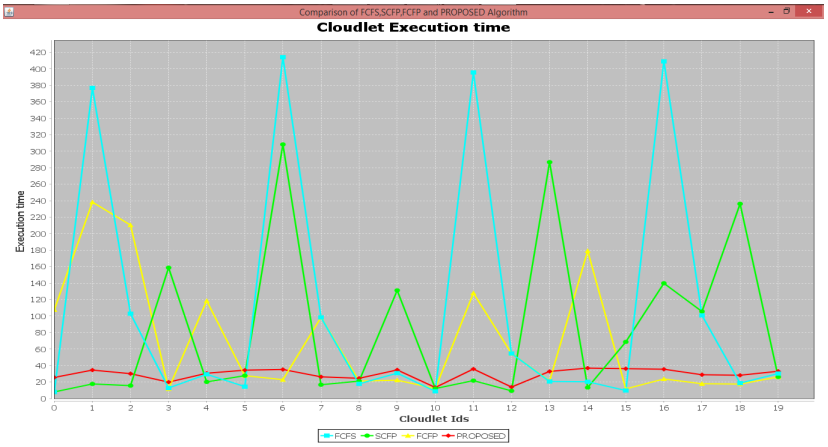

Figure 4.1: Graph for results of Experiment-1

Table 4.9: Results of Experiment-1

CLID	Size	FCFS	LCFP	SCFP	Proposed
0	2773	7.98	107.68	7.98	25.55
1	9706	376.9	238.3	17.66	34.49
2	6839	102.88	210.47	15.59	30.12
3	4087	12.78	11.76	158.71	19.86
4	8324	29.35	118.62	20.13	30.64
5	10405	14.45	27.71	27.71	34.38
6	11566	414.18	22.84	308.28	35.21
7	6130	98.41	98.41	16.69	26.28
8	6197	17.77	21.08	21.08	24.52
9	9175	30.75	22.26	131.18	34.66
10	3098	8.75	12.04	12.04	13.39
11	10347	395.57	127.95	21.71	35.8
12	2966	54.66	54.66	9.27	13.91
13	9354	20.64	21.25	286.81	32.86
14	5229	20.32	179.21	13.28	36.87
15	3726	9.69	11.65	68.67	36.17
16	11045	409.13	23.72	139.8	35.52
17	6393	100.83	17.91	105.54	28.82
18	6748	18.6	17.5	236.2	28.21
19	8815	30.3	26.16	26.16	33.06
Total		2173.94	1371.20	1644.5	590.32

4.6.2 Experiment 2

This experiment is carried out with 10 VMs and 50 Cloudlets as inputs. The configuration of virtual machines are given in Table 4.10 and the configuration of the cloudlets has been given in Table 4.11. The final total time to execute all the cloudlets by each of the algorithm has been provided in Table 4.12. Figure 4.2 represents the graphical form of the results.

In this experiment proposed algorithm(SCHFMC) performed much better than other algorithm. Here proposed algorithm(SCHFMC) improved the total execution by upto 39% of the FCFS algorithm, 37% of the SCFP algorithm and upto 24% improvement over FCFP algorithm. This experiment explains that proposed algorithm(SCHFMC) performs in much better way as number of cloudlets increases.

Table 4.10: VM list with MIPS capacity for Experiment-2

VMID	0	1	2	3	4	5	6	7	8	9
MIPD	1095	24	796	1973	346	1802	1923	1794	1878	961

Table 4.11: Cloudlet list with MI length for Experiment-2

CLID	**MI**	**CLID**	**MI**	**CLID**	**MI**	**CLID**	**MI**	**CLID**	**MI**
0	3470	10	7518	20	7565	30	4262	40	8952
1	2929	11	2554	21	4862	31	7039	41	6738
2	8050	12	8122	22	5768	32	2469	42	1863
3	7713	13	1571	23	6900	33	3756	43	6654
4	4442	14	8357	24	5135	34	1347	44	4043
5	4850	15	7743	25	2935	35	4186	45	2727
6	1451	16	7034	26	2219	36	4755	46	4597
7	3536	17	3391	27	6846	37	7428	47	6269
8	6695	18	2037	28	8943	38	4157	48	8186
9	4099	19	7921	29	6914	39	6736	49	5904

Table 4.12: Results of experiment 2

Algorithm	**FCFS**	**FCFP**	**LCFP**	**Proposed**
Total Execution Time	4836.45	4270.7	5944.5	802.87

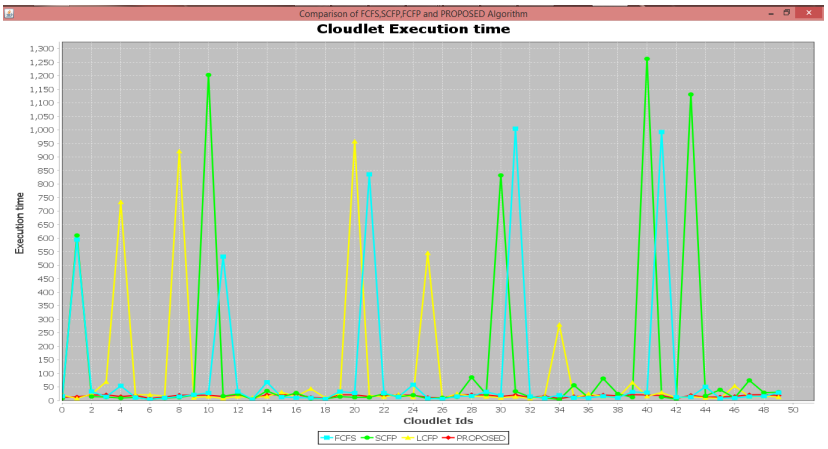

Figure 4.2: Graph for results of experiment 2

4.6.3 Experiment 3

In this experiment, Series of random variations has been studied. It is consist of 5 experiments and its graphical representation each with 10 VMs and 50 Cloudlets. Graphical results has been shown from Figure 4.3

MI of each cloudlet lies some random value between 1000 to 9000 and

MIPS of each VM lie some random value between 0 to 2000

One experiment with 20 VMs and 200 Cloudlets has been carried out and as prevous experiment proposed algorithm(SCHFMC) performs much better than other algorithms. Results has been shown in last image of Figure 4.3.

In multiple simulation with 20 VMs and 200 Cloudlets proposed algorithm(SCHFMC) has improved total execution time of all 200 Cloudlets by 25% over FCFS algorithm, 27% over SCFP and over 21% over LCFP algorithm.

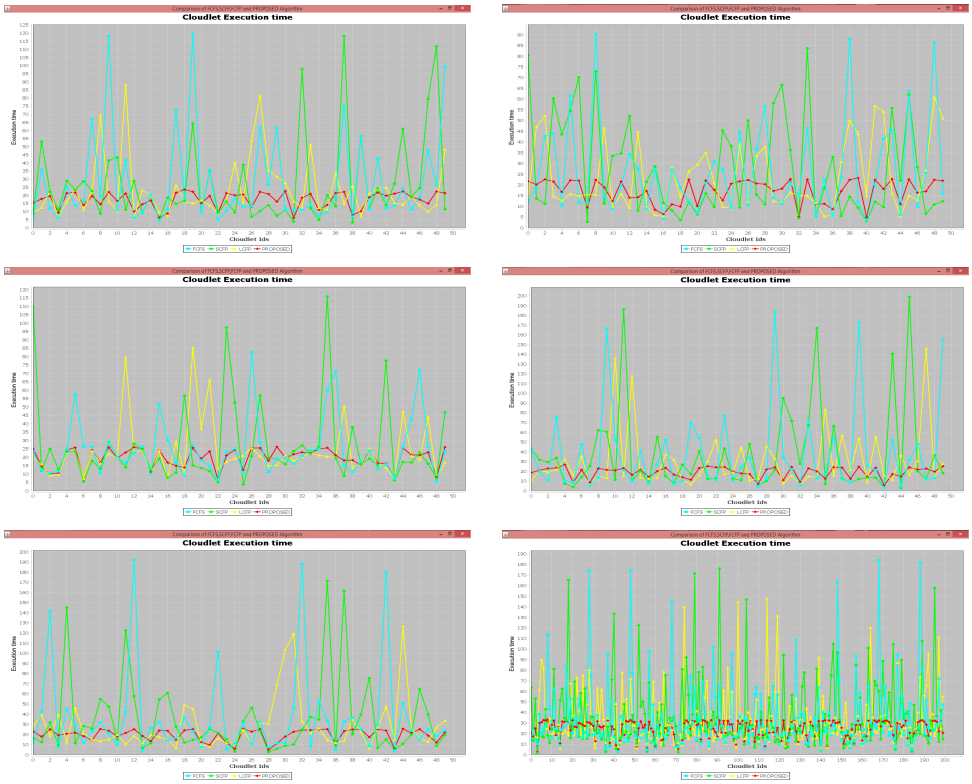

Figure 4.3: Graph for results of Experiment 3

4.7 Conclusion

This proposed algorithm(SCHFMC) has proper load balancing mechanism which improved makespan of each VM significantly. Moreover overall task completion time also decreases gradually. Multiple experiment with various input has been carried out and in all it perform very well. In all three experiments proposed algorithm has improved total execution time of all the cloudlets by VMs by 26% over FCFS assignment, 30% over SCFP scheduling while 19% improvement over LCFP scheduling. Table 4.13 shown the performance of the proposed algorithm(SCHFMC) compared to other algorithm as found in previous experiments.

Table 4.13: Performance of Proposed Algorithm(SCHFMC)

Algorithms	**Experiment-1**	**Experiment-2**	**Experiment-3**	**Average**
FCFS	15	38	25	26
FCFP	25	37	27	30
LCFP	11	25	21	19

Chapter 5

Conclusion

5.1 Conclusion

Task Scheduling is one of the major issue in cloud computing environment. Because it is responsible for the execution of the tasks also better utilization of the resources is analysed by task scheduling methods. Task scheduling policies must take care that task is executed according to the priority set by user. If user want that his task must executed according to completion time of task than scheduler must take care of completion time by keeping in mind that utilization of resources like VMs improves. From the thesis it is clear that proposed task scheduling algorithms performs much better than algorithm discussed and analysed in previous chapter of thesis. Proposed algorithm takes care that load on each machine is equally distributed hence it has proper load balancing schemes. In the end total completion time of all task significantly decreased and make span of each VM is improved significantly.

The proposed algorithm of this thesis has proper load balancing strategies that is why it has performed very well. This algorithm is scales and gives same improvement. As increment of the cloudlets to the same number of virtual machines, it gives more better results and improved makespan of each virtual machine.

5.2 Scope for Future Research

In this work, fault tolerance is not considered. If some VM become faulty during allocation or during execution that proposed algorithm will start giving random results. Hence during allocation algorithm should also takes care of fault or some future error in machine. Fault tolerance can create more problem during execution and can increase cost of execution unnecessarily. Hence this algorithm can be further extended.

Bibliography

[1] Fox, Armando and Griffith, Rean and Joseph, Anthony and Katz, Randy and Konwinski, Andrew and Lee, Gunho and Patterson, David and Rabkin, Ariel and Stoica, Ion Above the clouds: A Berkeley view of cloud computing, Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 2009.

[2] Dikaiakos, Marios D and Katsaros, Dimitrios and Mehra, Pankaj and Pallis, George and Vakali, Athena ; Cloud computing: Distributed internet computing for IT and scientific research, Internet Computing, IEEE, 2009.

[3] Zhang, Qi and Cheng, Lu and Boutaba, Raouf ; Cloud computing: state-of-the-art and research challenges; Journal of internet services and applications, 2010.

[4] Peter Mell, Timothy Grance The NIST Definition of Cloud Computing, National Institute of Standards and Technology, Special Publication 800-145, 2009.

[5] Calheiros Rodrigo N and Ranjan Rajiv and Beloglazov, Anton and De Rose, Buyya Rajkumar ; CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms; Software: Practice and Experience, Vol 41, no 2, p23-50, 2011.

[6] Calheiros Rodrigo N and Ranjan Rajiv and Buyya Rajkumar ; Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services; arXiv preprint arXiv:0903.2525, 2009.

[7] Amalarethinam, DI George and Muthulakshmi, Palaniandy; An Overview of the scheduling policies and algorithms in Grid Computing; International Journal of Research and Reviews in Computer Science, vol 2, no 2, p280-294 2011.

[8] Sun, Hong and Chen, Shi-ping and Jin, Chen and Guo, Kai ; Research and simulation of task scheduling algorithm in cloud computing; TELKOMNIKA Indonesian Journal of Electrical Engineering, Vol 11, No 11, p6664-6672 2013.

[9] Bala, Anju and Chana, Inderveer ; A survey of various workflow scheduling algorithms in cloud environment; 2nd National Conference on Information and Communication Technology (NCICT), pages 26-30, 2011.

[10] Tilak, Sujit and Patil, Dipti ; A survey of various scheduling algorithms in cloud environment; International Journal of Engineering Inventions, Vol 1, No 2, P36-39, 2012.

[11] Chawla, Yogita and Bhonsle, Mansi ; A study on scheduling methods in cloud computing; International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Vol 1, No 3, p12–17, 2012.

[12] Wu, Xiaonian and Deng, Mengqing and Zhang, Runlian and Zeng, Bing and Zhou, Shengyuan ; A task scheduling algorithm based on QoS-driven in cloud computing; Procedia Computer Science, Vol 17, P1162-1169 2013.

[13] Tripathy, Lipsa and Patra, Rasmi Ranjan; Scheduling in cloud computing; International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol 4, No 5, p21–7, 2014.

[14] Wu Ju-Hua; Research of Resource Allocation in Cloud Computing Based on Improved Dual Bee Colony Algorithm; International Journal of Grid Distribution Computing, Vol. 8, No 5, p117-126, 2015.

[15] Agarwal, Dr and Jain, Saloni and others ; Efficient optimal algorithm of task scheduling in cloud computing environment; arXiv preprint arXiv:1404.2076, 2014.

[16] Singh, Lal Shri Vratt and Ahmed, Jawed ; A GREEDY ALGORITHM FOR TASK SCHEDULING & RESOURCE ALLOCATION PROBLEMS IN CLOUD COMPUTING; International Journal of Research & Development in Technology and Management Science–Kailash, Vol 21, No 1, 2014.

[17] Chang, Fangzhe and Ren, Jennifer and Viswanathan, Ramesh ; Optimal resource allocation in clouds; Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, p418-425, 2010.

[18] Reddy, J Geetha and others ; A Review Work On Task Scheduling In Cloud Computing Using Genetic Algorithm; International Journal of Technology Enhancements and Emerging Engineering Research, Vol 2, No 8, P241–245, 2013.

[19] Shimpy, Er and Sidhu, Mr Jagandeep ; Different Scheduling Algorithms In Different Cloud Environment; algorithms, Vol 3, No 9, 2014.

[20] Gulati, Ajay and Chopra, Ranjeev K ; Dynamic round robin for load balancing in a cloud computing; International Journal of Computer Science and Mobile Computing, Vol 2, No 6, p274–278, 2013.

[21] Guo, Lizheng and Zhao, Shuguang and Shen, Shigen and Jiang, Changyuan ; Task scheduling optimization in cloud computing based on heuristic algorithm; Journal of Networks, Vol 7, No 3, p547–553, 2012.

[22] Li, Kai and Wang, Yong and Liu, Meilin ; A Task Allocation Schema Based on Response Time Optimization in Cloud Computing; arXiv preprint arXiv:1404.1124, 2014.

[23] Bhoi, Upendra and Ramanuj, Purvi N ; Enhanced max-min task scheduling algorithm in cloud computing; International Journal of Application or Innovation in Engineering and Management, Vol 2, No 4, p259–264, 2013.

[24] Elzeki, OM and Reshad, MZ and Elsoud, MA ; Improved max-min algorithm in Cloud computing; International Journal of Computer Applications, Vol 50, No 12, 2012.

[25] Liu, Gang and Li, Jing and Xu, Jianchao ; An improved min-min algorithm in cloud computing; Proceedings of the 2012 International Conference of Modern Computer Science and Applications, p47–52, 2013.

Zhao, Chenhong and Zhang, Shanshan and Liu, Qingfeng and Xie, Jian and Hu, Jicheng; Independent tasks scheduling based on genetic algorithm in cloud computing; Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference , p1–4, 2010.

[26] Shah, MR Manan D and Kariyani, MR Amit A and Agrawal, MR Dipak L; Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm; IJCSITS), ISSN, p2249–9555, 2013.

[27] Das, Pranesh and Khilar, Pabitra Mohan ; LBVFT: A Load Balancing Technique for Virtualization and Fault Tolerance in Cloud Computing; International Journal of Computer Applications, Vol 69, No 28, p14-18, 2010.

[28] Kaur, Rajwinder and Luthra, Pawan; Load Balancing in Cloud Computing; Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC, 2012.

[29] Katyal, Mayanka and Mishra, Atul ; A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment; arXiv preprint arXiv:1403.6918, 2014.

[30] Ma, Tinghuai and Chu, Ya and Zhao, Licheng and Ankhbayar, Otgonbayar ; Resource allocation and scheduling in cloud computing: Policy and algorithm; IETE Technical Review, Vol 31, No 1, p4–16, 2014.

[31] Piraghaj, Sareh Fotuhi and Calheiros, Rodrigo N and Chan, Jeffrey and Dastjerdi, Amir Vahid and Buyya, Rajkumar ; Virtual Machine Customization and Task Mapping

Architecture for Efficient Allocation of Cloud Data Center Resources; The Computer Journal, 2015.

[32] Somani, Rajkumar and Ojha, Jyotsana ; A Hybrid Approach for VM Load Balancing in Cloud Using CloudSim; International Journal of Science, Engineering and Technology Research (IJSETR), Vol 3, No 6, 2014.

[33] Ray, Soumya and De Sarkar, Ajanta ; Execution analysis of load balancing algorithms in cloud computing environment; International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol 2, No 5, p1–13, 2012.

[34] Parsa, Saeed and Entezari-Maleki, Reza ; RASA: A new task scheduling algorithm in grid environment; World Applied sciences journal, Vol 7, p152–160, 2009.

[35] Rawat, Pradeep Singh and Saroha, GP and Barthwal, Varun ; Quality of service evaluation of SaaS modeler (Cloudlet) running on virtual cloud computing environment using CloudSim; International Journal of Computer Applications, Vol 53, No 13, 2012.