MFL-APSO AND ADRS-DDMC BASED ADVANCED ANTICIPATORY PERFORMANCE IMPROVEMENT MODEL FOR CLOUD COMPUTING

A Thesis

Submitted for the award of the degree of

Doctor of Philosophy

In
Computer Science & Engineering

Submitted to

Maharishi University of Information Technology, Lucknow (UP)

Submitted By-Umesh Kumar Lilhore

Under the Supervision of

Dr. Santosh Kumar

Head Dept. of CSE

Year: 2016-17

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHARISHI UNIVERSITY INFORMATION TECHNOLOGY,

LUCK NOW (UP)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the research work embodied in the thesis entitled **MFL-APSO and ADRS-DDMC Based Advanced Anticipatory Performance Improvement Model for Cloud Computing,** by **Mr. Umesh Kumar Lilhore** in the faculty of **Computer Science & Engineering**, for the Degree of **Doctor of Philosophy**, was carried out under my supervision and that the candidate has worked under me for the period required under the Ordinance.

Date -

Dr. Santosh Kumar

(Research Guide & Head)

Dept. of Commuter Science & Engineering

Maharishi University Information Technology

Lucknow (UP)

MAHARISHI UNIVERSITY INFORMATION TECHNOLOGY,

LUCK NOW (UP)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CANDIDATE'S DECLARATION

I declare that the thesis entitled **MFL-APSO and ADRS-DDMC Based Advanced Anticipatory Performance Improvement Model for Cloud Computing** is my own work conducted under the supervision of **Dr. Santosh Kumar (Supervisor)** at **Maharishi University Information Technology Lucknow (UP),** approved by Research Degree Committee. I have put in more than 200 days/600 Hrs. of attendance with a supervisor at the center.

The content embodied in this Ph.D. thesis has not been submitted for the award of any other degree/diploma. I declare that I have faithfully acknowledged, given credit to and referred to the research workers wherever their works have been cited in the text and the body of the thesis. I further certify that I have not deliberately used any other person's work, in form of para, text, data, results, etc. reported in elsewhere (the journals, books, magazines, reports, dissertations, theses, etc., or available at websites/ Internet) and included them in this Ph. D. thesis and cited that as my own work.

Date: ……..

Place: Lucknow

Umesh Kumar Lilhore

MAHARISHI UNIVERSITY INFORMATION TECHNOLOGY,

LUCK NOW (UP)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIVA VOCE CERTIFICATE

This is to certify that research work embodied in this entitled **MFL-APSO and ADRS-DDMC Based Advanced Anticipatory Performance Improvement Model for Cloud Computing** Submitted to the **Department of Computer Science & Engineering, Maharishi University Information Technology Lucknow (UP),** in fulfillment of the requirement for the degree of **Doctor of Philosophy** in **Computer Science & Engineering** has been approved after an oral examination of the same in collaboration with external examiner.

External Examiner

Date-

Place: Lucknow

Head

Dept. of Commuter Science & Egg.

Maharishi University Information Technology

Lucknow (UP)

ACKNOWLEDGEMENT

With due respect, I express my deep sense of gratitude to my respected and learned guide **Dr. Santosh Kumar, Head Department of Computer Science & Engineering at Maharishi University Information Technology Lucknow (UP),** for giving me the right direction, excellent references and taking a keen interest in the research work for PhD, right from the beginning. I am thankful to him for the encouragement and guidance provide for completion of this dissertation.

I am also grateful to respected **Dr. Madhulika Singh, Dean Academic of Research Department at Maharishi University Information Technology Lucknow (UP),** for permitting me to utilize all the necessary facility of the college. I am thankful to all the teacher of the department for their support and encouragement. I must acknowledge the academic resources that I have got from the **Maharishi University Information Technology Lucknow (UP).** I would like to thanks, administrative and technical staff member of the department who have been kind enough to advice and help.

However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them. I would like to express my gratitude towards my lovely parents, my son and my wife for their kind co-operation and encouragement which help me in the completion of this dissertation report.

Umesh Kumar Lilhore

ABSTRACT

Cloud computing is an improved form of previous IT based technologies such as Grid and Cluster computing. It's a widely used technology over the internet, which serves computing resources such as Infrastructure, Platform and Software, to cloud user on pay and use basis. Cloud computing allows optimum utilization of computing resources at a lower cost without any compromises in performance. Cloud services and its consumers are increasing rapidly. Now it's more important for cloud service provider to maintain better services and cloud performance. Cloud computing provides powerful computing power for large-scale application across global locations. Cloud computing structures are based on a multi-tenancy technique. A better cloud computing architecture requires higher availability and efficiency of the computing resources.

In cloud computing, better performance is always a challenging and sensitive issue. The performance of cloud computing depends on the various factors such as application performance, network performance, cloud infrastructure performance and geographical environment. However, a cloud user always demands the higher speed network performance to support heavy computing applications and better experience and hence is more concerned about the network performance.

To determine the cloud performance factors, firstly we have analyzed various previous research works and their publications related to performance improvement of cloud computing. To improve the performance of cloud system, optimizations of resources are required at both the hardware and software level. The hardware-based changes are more complex and costly as compared to software-based changes. Software-based changes include input and output speed between the various computing layers, load balancing policies and broker policies.

In cloud computing, load balancing is one of the interesting and prominent research topics, which has gained large attention recently. The load balancing methods balance the various incoming loads and distribute into different cloud data centers. After storing user data in the data center, the load balancing method replicate the data into multiple virtual machines to avoid

any data loss. For efficient load balancing, many algorithms and approaches are proposed by various researchers with the aim of balancing the overall workload. This research work presents a software-based performance optimization method by efficient load balancing between various cloud layers. This research work investigates Advance Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC). The proposed AAP-IMC performance improvement model presents a set of solutions for load balancing in the cloud. The proposed model uses a combined strategy of two proposed methods MFL-APSO and ADRS-DDMC in various phases.

First phase is based on Modified Fuzzy Logic and Advanced Particle Swarm Optimization Model (MFL-APSO) to optimize the total execution time of tasks in the workflow applications. The main key objective of applying the MFL-APSO method is to minimize the total tasks execution time by verifying the load fluctuations of the interconnected tasks. The variance of the algorithm considers factors such as load fluctuations and optimization of the data retrieval time. The proposed model MFL-APSO is validated by applying five workflow structures with different data block sizes. The experimental results are compared with Heterogeneous Earliest Finish Time (HEFT) algorithm and Scalable Heterogeneous Earliest Finish Time (SHEFT).

Second phase is based on Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud computing (ADRS-DDMC). This phase introduces a novel dynamic data replication method that is functioning based on anticipations to create the pre-replicas for future needs of the sites. The method optimizes load balancing by increasing the data availability among the existing sites. The experimental results are compared with Least Recently Used (LRU) and Least Frequently Used (LFU).

Final phase is based on a novel Advance Anticipatory Performance Improvement Model for Cloud computing (AAP-IMC) which is based on models of phase one and phase two. Proposed AAP-IMC and existing methods are implemented in Cloud-Sim simulator with Java programming language. The experimental results of the proposed method are compared with results of phase-1, phase-2 and existing Honeybee, LRU, LFU, HEFT and SHEFT methods and various performance comparisons parameters are calculated such as execution time, response

time, resource utilization, workflow, software error rates and data transfer rate. Experiment results clearly show that proposed AAP-IMC performs outstandingly as compared to the existing method.

Keywords- *Cloud computing, performance of cloud computing, load balancing MFL-APSO, ADRS-DDMC and AAP-IMC*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBREVIATIONS	DESCRIPTION
APSOM	Advanced Particle Swarm Optimization Model
AMAZON EC-2	Amazon Elastic Cloud Computing
ANM	Anisotropic Network Model
API	Application Programming Interface
AAP	Advanced Anticipatory Performance
ADRS	Anticipatory Data Replication Strategy
BMHA	Batch Mode Heuristic Scheduling
CC	Cloud Carrier
CLC	Cloud Controller
CSP	Cloud Service Provider
DAG	Directed Acyclic Graph
DBC	Deadline and Budget Constraint
DCP	Dynamic Critical Path
DCWLB	Deadline Constrained Workflow Load Balancing
DDM	Dynamic Distributed Model
DPSO	Discrete Version of PSO
ENM	Elastic Network Models
FCFS	First Come First Served
FMOC	Finite Multi-Order Context
GA	Genetic Algorithm
GBEST	Global Best Position
GPU	Graphical Processing Unit
GRMS	Global Replica Management System
HBLM	Honey Bee Load Balancing
HEFT	Heterogeneous Earliest Finish Time

HTML	Hyper Text Mark-up Language
I/O	Input Output
IMC	Improvement Model for Cloud Computing
IaaS	Infrastructure as a Service
ICT	Information & Communication Technology
IoT	Internet of Things
IT	Information Technology
JDK	Java Development Kit
LFU	Least Frequently Used
LRU	Least Recently Used
Make span	Total Length of Schedule
MBIT/S	Megabit per Second
MFL	Modified fuzzy Logic
NIST	National Institute of Standards and Technology
NP Problem	Non-Deterministic Polynomial Time
PaaS	Platform as a Service
PBEST	Best Local Position
PSO	Particle Swarm Optimization
PSHOW	PSO Based Heuristic for Workflow
QoS	Quality of Service
QHWLB	QoS Heuristic Workflow Load Balancing
RC	Relative Cost
RDPSO	Revised Discrete Particle Swarm Optimization
RR	Round Robin
SaaS	Software as a Service
SDDR	Smart Dynamic Data Replication in Cloud
SHEFT	Scalable Heterogeneous Earliest Finish Time
SLA	Service Level Agreement

TICC	Transaction Intensive Cost Constrained
URL	Uniform Resource Locator
VM	Virtual Machine
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

In this chapter, a high-level overview of the complete research work is described. It covers the objective of the research, problem statement, research contributions and research questions. This chapter also describes expected the outcome of the work. In the last concludes an outline overview of the complete existing chapters.

1.1 CLOUD COMPUTING

Cloud computing is a new and innovative computing paradigm, in which a pool of large variety systems are connected in private, public or hybrid networks, to provide various services such as dynamically scalable computing infrastructure for application development, storage of data and file. With the innovation of cloud technology, the cost of application hosting, content storage, computation and delivery costs and time reduced significantly. Figure 1.1 shows basic cloud computing model with components. These include resource management and load balancing, which are developed and executed according to the user's requirements.

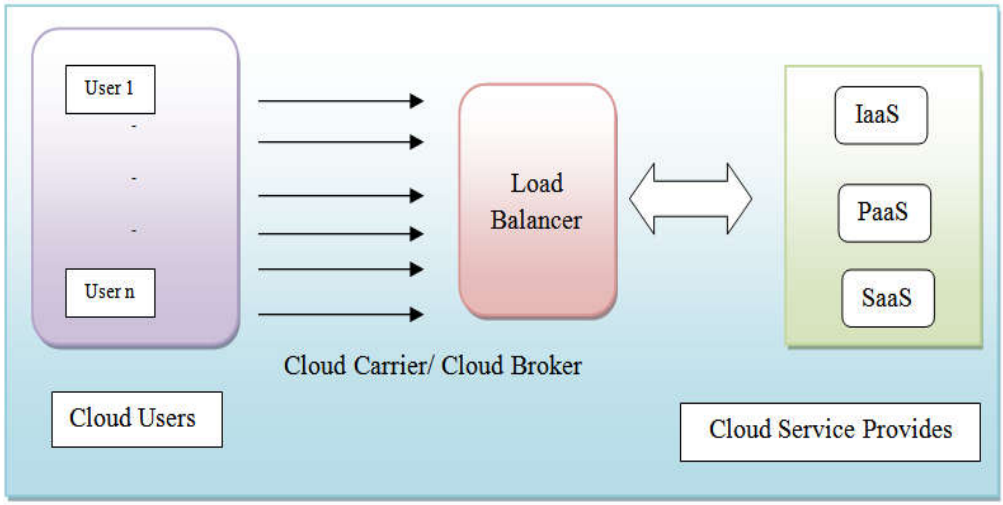

Figure 1.1 Basic Models of Clouds Computing

A number of cloud computing experts and researcher have suggested a variety of definitions and views about of cloud computing. Cloud computing can be defined (Alicherry. M, et al., 2013) as-

A cloud computing is a large scale based distributed computing technology or paradigm, that is widely driven by economies of scale, in which a pool of computing services are delivered to external cloud users, on demand pay and use basis over the internet, services such as abstracted, virtualized, dynamically scalable managed computing power, storage, platform.

1.1.1 Cloud Components-A cloud system consists of three types of major components. Cloud components are a various data center, clients and distributed servers (Daeyong Jung, et al., 2013). In the cloud, each element has a definite and important purpose and plays a specific role (figure 1.1.1.1 shows cloud computing model).

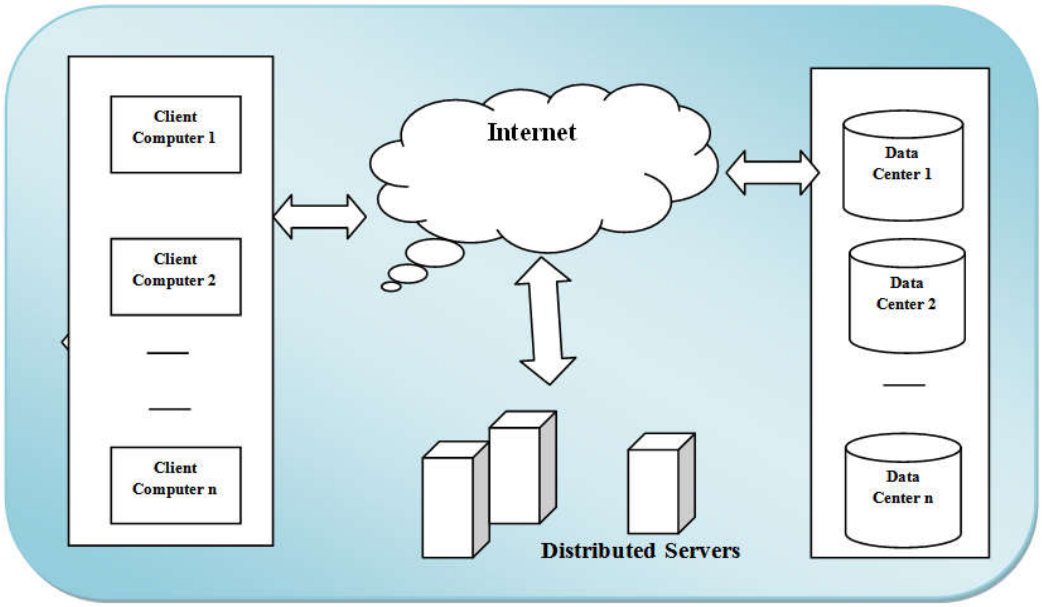

Figure 1.1.1.1 Components of Cloud Computing Model

Major cloud components are as follows:

I. Data Center-A data center is a collection of servers hosting different types of applications. An end user can directly connect and access data and various applications from the data center. As per Dodding Probhuling, et al., a data center may take place at a long or near distance from the clients. Now, these days a concept called virtualization is also used, to install application software's, which allows multiple instances of virtual server applications.

- **Key Requirements of Data Center-**Egwutuoha, et al., 2014, described that data center infrastructure should meet the following attributes to ensure that data is accessible to users

effectively and efficiently all the time. Figure 1.1.1.2 shows essential key requirements for a data center.

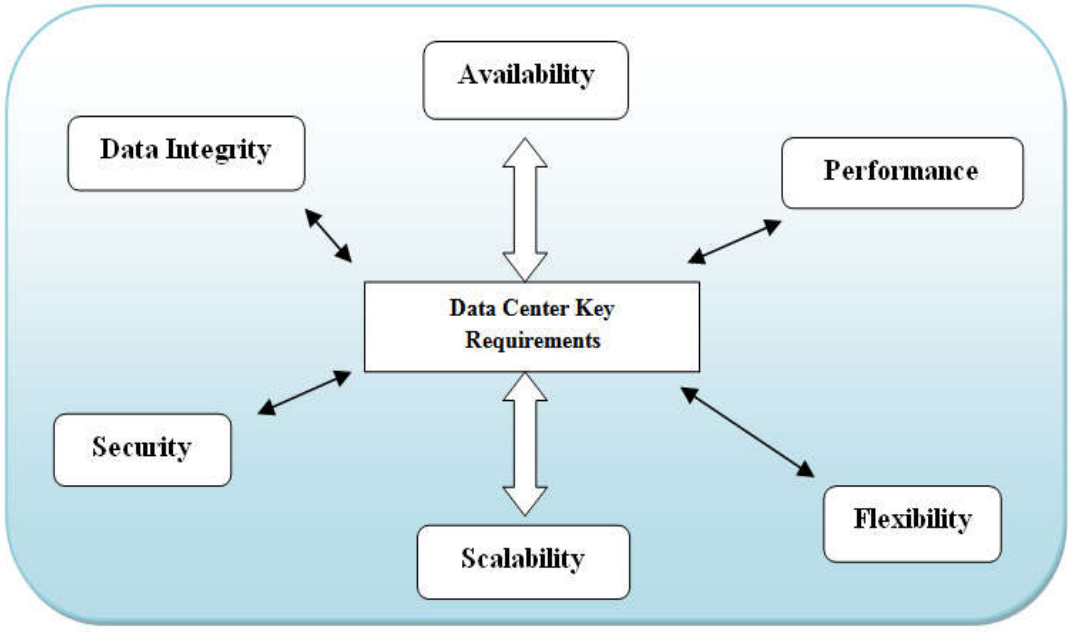

Figure 1.1.1.2 Key Requirements of a Data Center

a) **Availability-**All data center elements should be planned to assure availability. Availability plays an important role in cloud performance. The unavailability of data for a cloud customer can have a negative impact on a cloud business.
b) **Performance-**In cloud computing better performance is always a desirable task. Cloud data center and other cloud computing resources should provide high performance and service to all processing requests of the client with high speed.
c) **Scalability-**It is the most desired need for cloud data center. A cloud data center should allocate more processing of storage space without interrupting business operations.
d) **Manageability-**Manageability is always important for cloud computing. It can be achieved by using reduction and automation of manual intervention in common tasks.
e) **Security-**Data security is also a major concern in cloud computing. It can be achieved by cryptography and various data encryption techniques. A third party auditor also called TPA is responsible for data security. TPA maintains data security and data integrity.
f) **Data Integrity-**Data integrity ensures that data is not changed or modified and the data are stored in the original copy.

II. Cloud Clients-According to Fangzhe Chang Ren, J. et al., 2011, an end user directly interact with the various clients to manage cloud information in cloud computing client has following types:

a) **Thin Clients-**They do not perform any computation work. They can only display the particular information. Servers perform all the works for them. Thin clients system does not have any internal memory.
b) **Mobile Clients-**Mobile cloud applications move the computing power and data storage away from the mobile devices and into powerful and centralized computing platforms located in clouds, which are then accessed over the wireless connection based on a thin native client. Such as Windows Mobile, Smartphone like as Blackberry or I-Phone.
c) **Thick Clients-**These use different browsers based clients such as various browsers like as Internet Explorer, Safari, Mozilla Firefox, Netscape Navigator, Google Chrome to connect and access to the various Internet cloud.

III. Distributed Servers-Distributed servers are important to major parts of a cloud computing technology, which plays an important role in cloud computing environment and present by using the various Internet-based hosting applications (Faragardi, et al., 2013). However, at the time of using these applications from the cloud environment, the user must feel that he is using these applications from its own computer machine.

1.1.2 Characteristics of Cloud Computing-According to Gaochao Xu, et al., 2013, to qualify a technology as cloud computing NIST provide five essential characteristics:

a) **Rapid Elasticity-**Cloud computing provides the great elasticity capabilities by providing the unlimited resources. Cloud users can demand the computing resources, offered at the time of required and same amy is released once the requirement is over or completed.
b) **Network Access Broadly-**Capabilities available in the cloud can be accessed by a wide variety of devices. Such devices are tablets, laptops, smartphones and desktop computers, these devices can be used to access the resources.
c) **Measured Services-**In cloud computing the services utilized or consumed by the consumers must be measured for the billing purpose. Cloud computing systems provide the decision capability to measure how many quantities of computing resources used.

d) **Resource Pooling-**Cloud computing works on a distributed model where resources are distributed throughout the various data centers stored at various locations.

e) **On-Demand Self Service-**In cloud computing resources can be provisioned and de-provisioned as per the demand of the users.

1.1.3 Cloud Deployment Model-Cloud deployments are used to provide the services for a different kind of need (Gupta, et al., 2013). These deployment models are classified depending upon the scope of their accessibility i.e. within the organization, outside the organization or combination of these two. The four cloud deployment models defined by NIST are as follows:

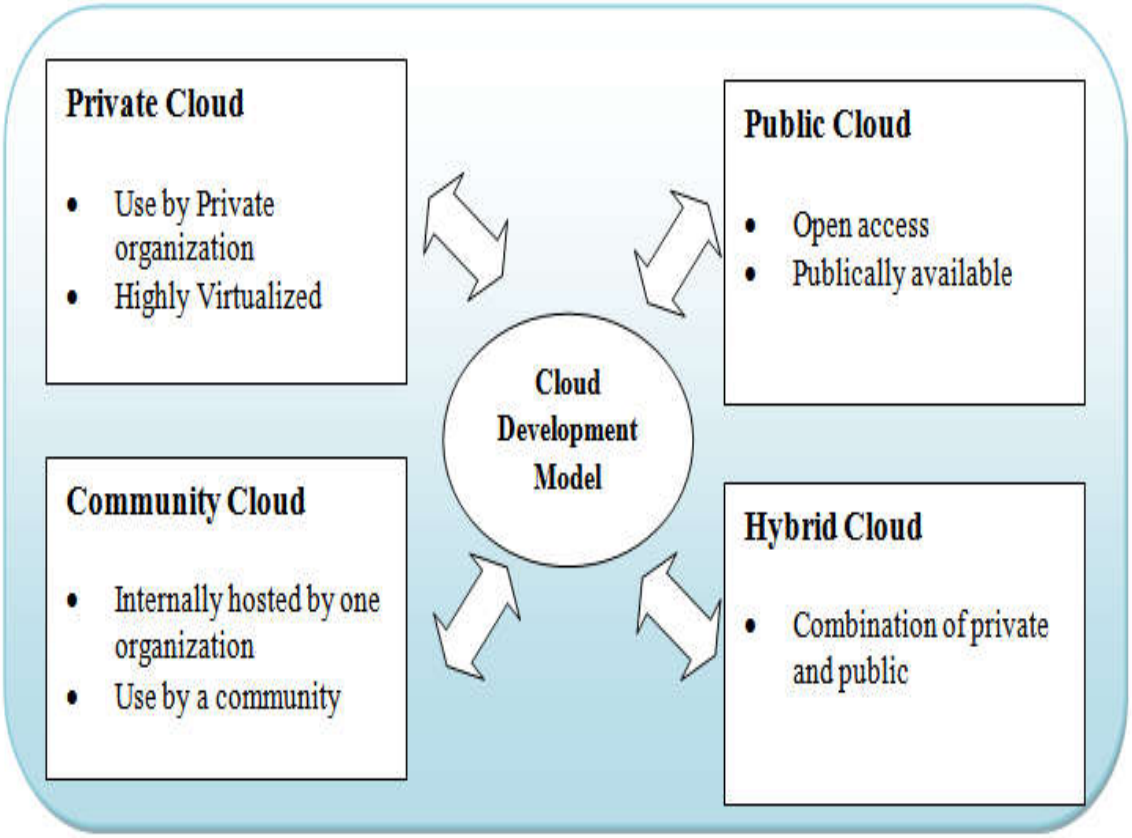

Figure 1.1.3 Cloud Deployment Model

1.1.3.1 Private Cloud-Private cloud is a deployment model (figure 1.1.3.1) that is solely maintained by the organization itself or by any third party level agreement between both. A private cloud is highly suitable where enterprises are having a multi-location presence.

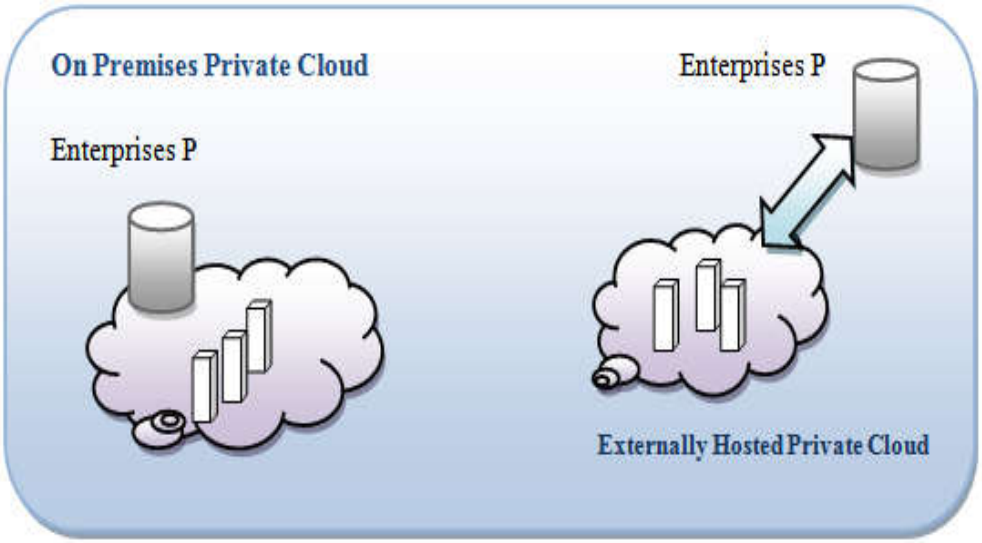

Figure 1.1.3.1 Private Cloud

1.1.3.2 Public Cloud-This model refers to a delivery model (figure 1.1.3.2) where a cloud service provider (CSP) provides massively scalable computing resources, such as CPU and storage capacity or software application for the general public service.

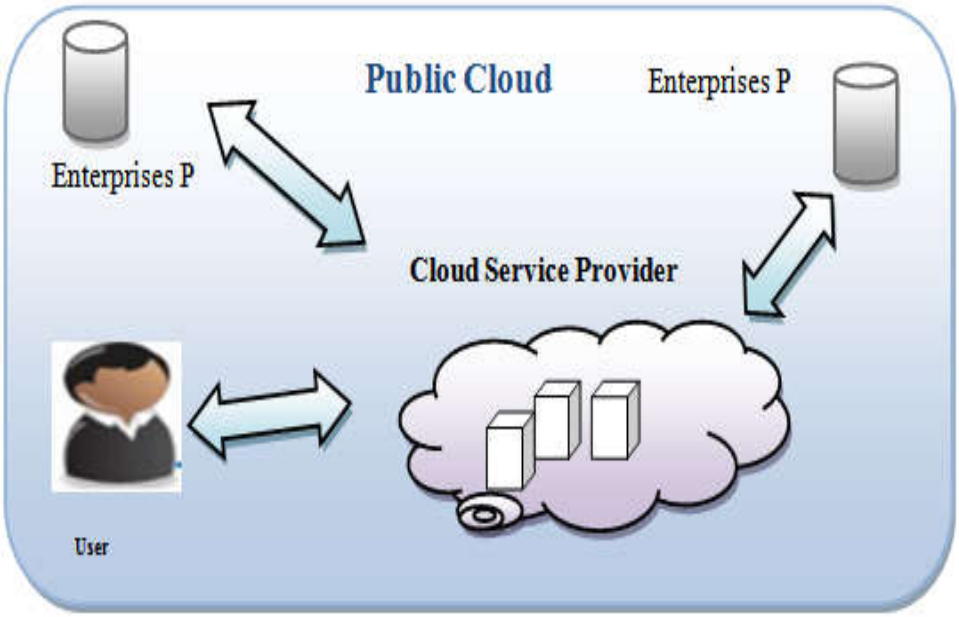

Figure 1.1.3.2 Public Cloud

1.1.3.3 Hybrid Model-The hybrid model is a combination of both private and public cloud, which blends the feature of both the private and public cloud models. Figure 1.1.3.3 shows hybrid model for cloud computing.

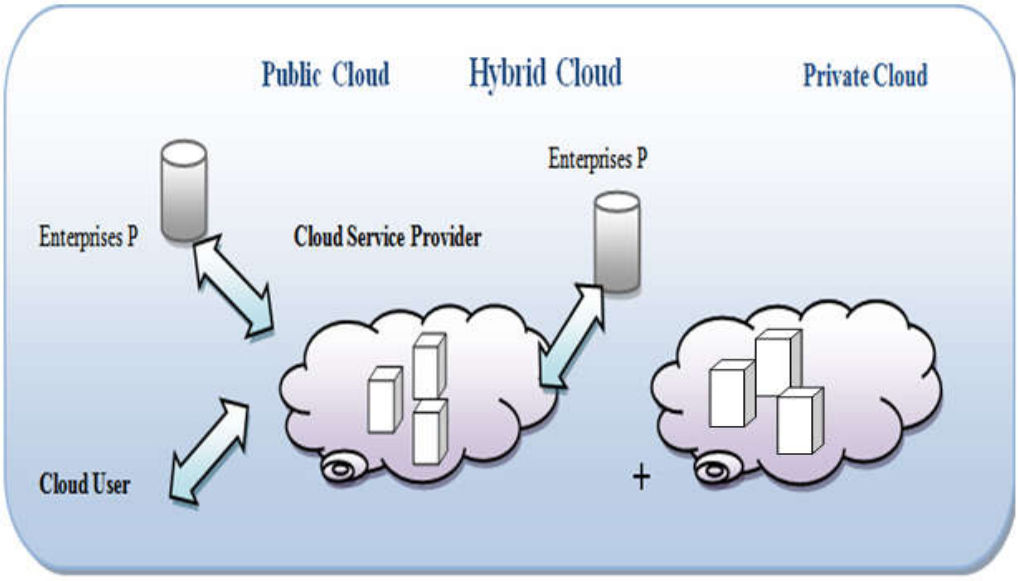

Figure 1.1.3.3 Hybrid Cloud

1.1.3.4 Community Cloud-Community cloud is used for more than one organization having similar requirements and shares the same context. In community cloud common characteristics are shared while other information's are confined to the owner organization's itself. It is a form of private cloud (figure 1.1.3.4).

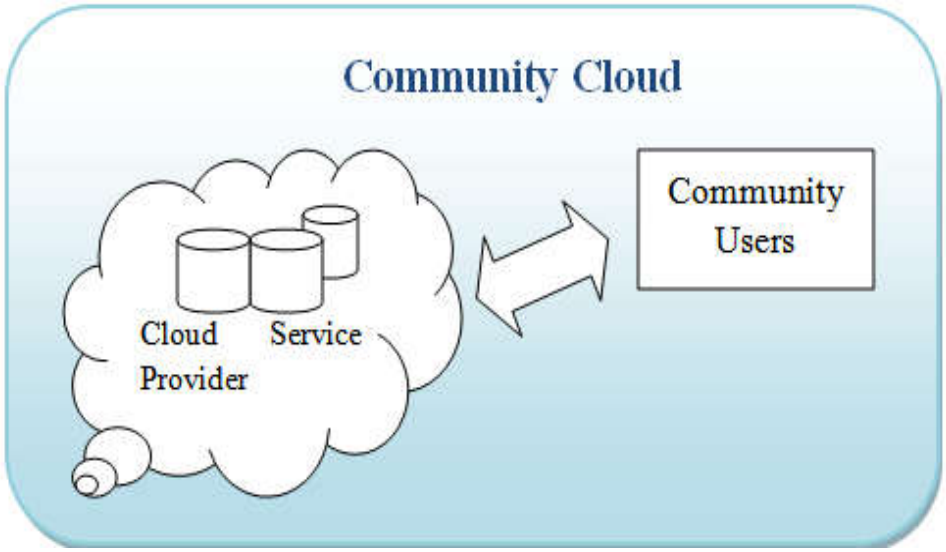

Figure 1.1.3.4 Community Cloud

1.1.4 Cloud Service Models-To cater the need of users, the cloud is offering different kinds of services. Cloud services have been categorized by NIST into following three categories:

1.1.4.1 Infrastructure as a Service (IaaS)-IaaS is generic computational infrastructure that is offered through the cloud. In this model resources such as a processor, memory and storage can be subscribed by the users and used on a pay as per the usage. Some of the major IaaS providers are Amazon EC-2, Amazon S-3, Eucalyptus, Go-Grid and Microsoft Live Mesh etc.

1.1.4.2 Platform as a Service (PaaS)-PaaS provides a more cost-effective way to develop and deliver the application. PaaS provides developer unlimited computing power. Thus, PaaS is driving a new era of innovation. Some of the major PaaS providers are Windows Azure, Heroku, Google App Engine and Engine Yard etc.

1.1.4.3 Software as a Service (SaaS)-SaaS is deployment models in which applications are hosted on cloud provider network and customers are accessing it using the internet. Some of the SaaS-based applications include customer resource management, collaboration, web analytics, invoices and office applications etc.

1.1.5 Challenges in Cloud Computing-However; we can say that many cloud-based services and cloud-oriented applications are not much efficient enough due to the following reasons:

1. Lack of information sharing.
2. Assumption of homogeneous environments and
3. The unpredictability of the environments.

Due to above reason, cloud-based technologies encounter with following major challenges:

Challenges in the Cloud Computing

Challenges	Description
Load Balancing	Improper load monitoring policies and load management for different cloud applications.
Energy Saving	It defines a standard metric for effective power usage and an efficient standard of infrastructure usage.
Security and Privacy	Lack of improved techniques in authorization and authentication for accessing the user's information.
Resiliency	The ability of the system to provide users with a standard level of

	services while experiencing faults and challenges in the system.
Reliability	The chance of failure in a standard period of time.
Resource Monitoring	Lack of accurate monitoring mechanism using sensors to collect the data from CPU load, memory load and etc.
Interoperability	Lack of standards for service portability between cloud providers.

Table 1.1.5.1 Challenges in the Cloud Computing

1.1.5.1 Advantages of Cloud Computing-Cloud computing offer plenty of advantages to the cloud users. Infinite availability of the resources, beneficial payment model and scalability, elasticity specifications of the cloud networks motivated businesses to change their legacy.

Advantages of Cloud Computing

Advantages	Description
Integrity	Data integrity can manage easily and avoid duplication.
Elasticity	Allocate and release resources upon usage.
Mobility	Accessing the cloud network is not dependent on time and location.
Disaster Recovery	Using the virtual backup's recovery will be faster by using.
Availability	Rapid deployment of the infrastructure to access the resources.
Low Infrastructure Cost	Helps small businesses to grow sooner.
Scalability	Add or remove resources from demands.
Increased Data Storage	Access to large storage capacity for data storage and a backup plan.

Table 1.1.5.2 Cloud Computing Advantages

1.2 CLOUD PERFORMANCE

In cloud computing, better performance is always desirable and challenging. The performance of the cloud system depends on various parameters; load balancing is one of them. Load balancing is a technique or method in which the complete or partial workload from one node is shifted to a respective free node on the other node in a network without disturbing the other running task (Mayank Mishra, et al., 2012). Load balancing is one of the main concerns in cloud computing

that has a major impact on defining the availability of the resources (figure 1.2). A proper load balancing strategy, tasks could be scheduled evenly among available resources which could lead to a balanced network. The lack of efficient load balancing tool can result in experiencing a long access time for users.

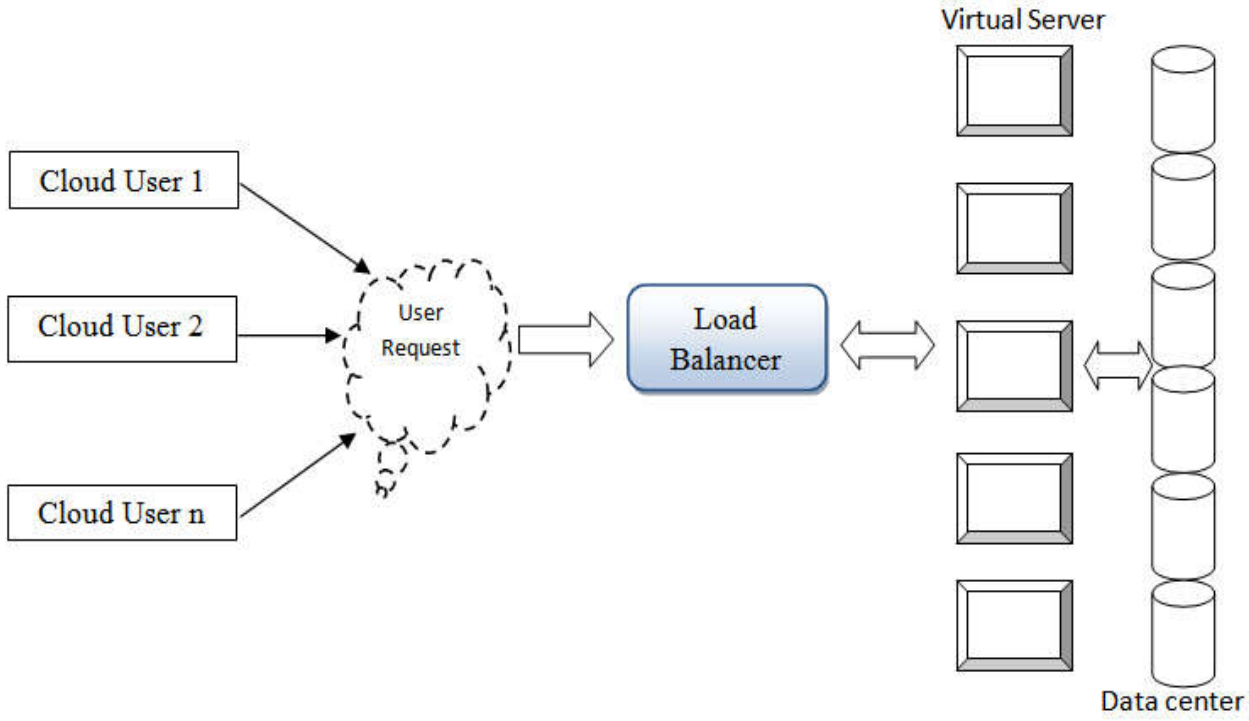

Figure 1.2 Load Balance in Cloud by Load Balancer

1.2.1 Why Load Balancing Important for Cloud Performance?-Load balancing provides following benefits:

a) Distributing a workload across multiple computers for improved performance.

b) The failure is avoided.

c) Improving the VMs to process efficiently.

d) Increasing consumption of system resources

e) Increase system throughput and performance.

1.2.2 Load Balancing Measurement Criteria-(Pawar C.S, et al., 2012) discussed the principle measurement criteria for load balancing cloud environment are given below:

a) **Throughput-** It is the total number of jobs, which have completed their execution on a given scale of time. It is required to have high throughput for better performance of the system.
b) **Fault Tolerance-**It is the ability to perform load balancing, by the correct algorithm efficiently without, any arbitrary link failure or node failure. Every load balancing algorithm should have a good fault tolerance approach or process.
c) **Migration Time-**The time that is required for a process to be transferred from one system to another system node for execution. Less migration time for a process shows better system performance.
d) **Response Time-**The time taken by a particular request to respond from the client and the server load balancing technique to respond. The minimum response time gives better performance.
e) **Resource Utilization-**This parameter gives information about resources utilized such as memory usage. For efficient load balancing in the system in cloud optimum resources should be utilized.
f) **Scalability-**It is the ability of the load balancing method, for a system with a finite number of processors and machines.
g) **Performance-**It is the overall efficiency of the cloud system. Higher performance is always desirable.

1.3 OBJECTIVES OF THE THESIS

Cloud computing provides powerful computing power for large-scale application across global locations. Cloud computing structures are based on a multi-tenancy technique. A better cloud computing architecture requires higher availability and efficiency of the computing resources. In cloud computing, better performance is always a challenging and sensitive issue. A complete performance of cloud computing systems depends on the various factors such as application performance, network performance, cloud infrastructure performance and environment.

However, a cloud user always demands the higher speed network performance to support heavy computing applications and better experience and hence is more concerned about the network performance. To determine the cloud performance factors, firstly we have analyzed various previous research works and their publications related to performance improvement of cloud

computing. To improve the performance of cloud system, optimizations of resources are required at both the hardware and software level. For performance improvement in cloud system hardware based changes are more complex and costly as compared to software-based changes. Software-based changes include input and output speed between the various computing layers, load balancing methods and broker policies. In the field of cloud computing, there are various open issues for researchers, load balancing is one of them, which has gained large attention recently. The load balancing methods balance the various incoming loads and distribute it to different cloud data centers. After storing user data in the data center, the load balancing method replicate the data into multiple virtual machines to avoid any data loss. For efficient load balancing, many algorithms and approaches are proposed by various researchers, with the aim of balancing the overall workload. This research work presents a software-based performance optimization method by efficient load balancing in between various cloud layers.

1.4 MAIN RESEARCH CONTRIBUTION

This research work investigates **Advanced Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC).** The proposed AAP-IMC performance improvement model presents a set of solutions for load balancing in the cloud. The proposed model uses a combined strategy of two proposed methods MFL-APSO and ADRS-DDMC, in various phases.

Phase-I: This first phase covers **Modified fuzzy Logic and Advanced Particle Swarm Optimization Model (MFL-APSO)** to optimize the total execution time of tasks in the workflow applications. The key objective of applying the MFL-APSO method is to minimize the total tasks execution time by verifying the load fluctuations of the interconnected tasks. The variance of the algorithm considers factors such as load variations and optimization of the data retrieval time. The proposed model MFL-APSO is validated by applying different five workflow structures with different data block sizes. The results are compared with Heterogeneous Earliest Finish Time (HEFT) algorithm and Scalable Heterogeneous Earliest Finish Time (SHEFT).

Phase-II: This second phase covers **Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing (ADRS-DDMC).** This phase introduces a novel dynamic data replication method that is functioning based on anticipations to create the pre-

replicas for future needs of the sites. The proposed ADRS-DDMC method optimizes load balancing by increasing the data availability among the existing sites. The results are compared with Least Recently Used (LRU) and Least Frequently Used (LFU).

The main contribution of the present research work is:

1. **First-**To develop an efficient heuristic methodology based on advance fuzzy logic for task load balancing.
2. **Second-**To develops an anticipatory performance model by data replication and dynamic distributed methods for cloud load balancing.
3. **Third-**To combines the above two strategies of Phase I and Phase-II and developed an Advanced Anticipatory Performance Improvement Model, for cloud computing. This proposed method uses a novel dynamic data replication method that is functioning based on Anticipations to create the pre-replicas for future needs of the sites by using an efficient heuristic methodology based on advanced fuzzy logic for task load balancing.

1.5 RESEARCH QUESTIONS

Cloud computing is a new technology, which is based on virtualizations. Cloud computing serves computing resources over the network to its user, which attract users to utilizes its service. This quality of cloud computing increases its number of users and computing resources.

The main research question which has been covered in this research work is:

- How to increase the throughput of cloud system?
- How to improve load balancing in the cloud for better performance?
- How to improve the cloud performance?
    - Apply the importance and path of the load between interconnected tasks for optimizing the weight balancing behavior in cloud computing.
- How to improve the availability of computing resources?
    - Pre-replicate the data with high uses in neighborhood servers before provisioning requests had been submitted.
    - Increase the availability of data.
    - Increase the efficiency of accessing.

Cloud computing provides optimum utilization of computing resources for cloud users as well as cloud service providers. Cloud users are demanding 24*7 error free services from CSP. Existing cloud computing system encounters several issues. This research mainly covers few of them. Below figure 1.5 describes major challenges in cloud performances (Xin Li, 2014).

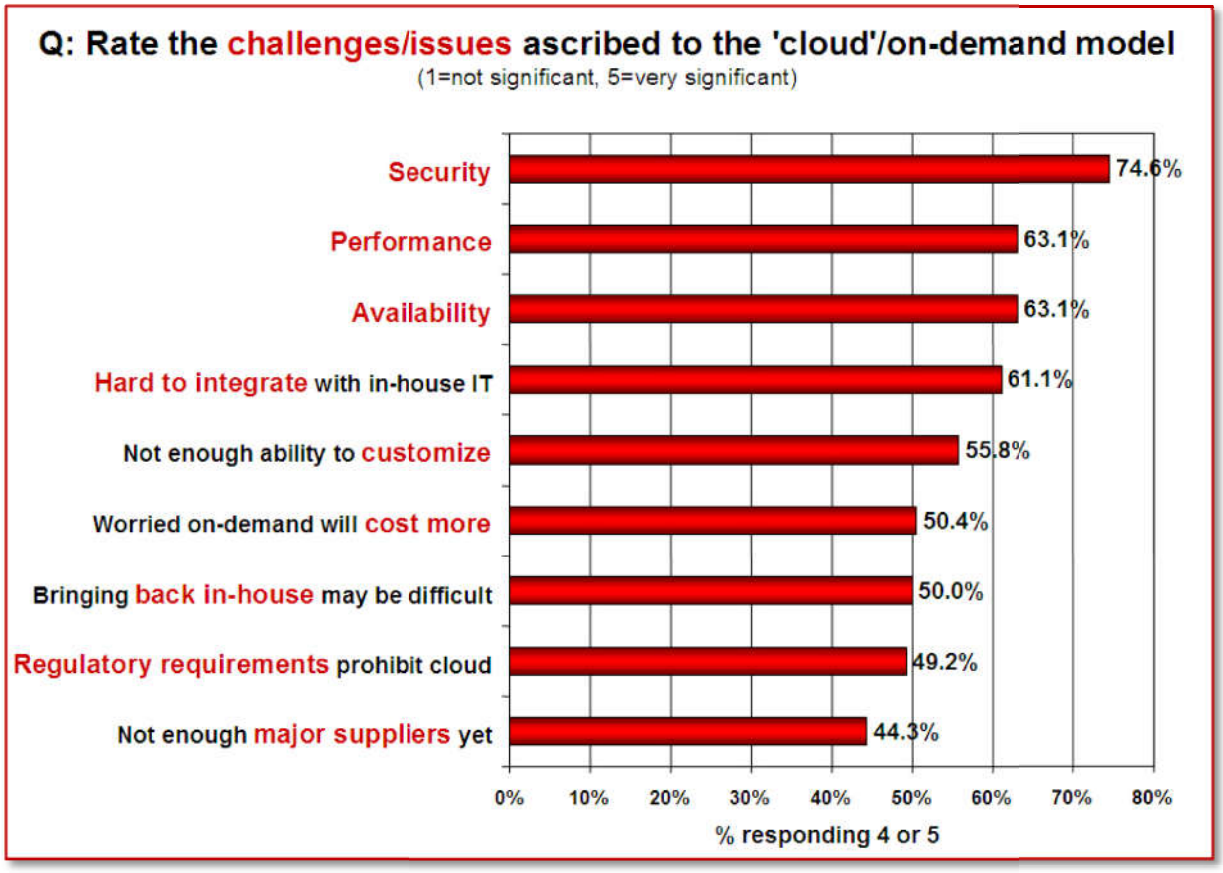

Figure 1.5 Major Challenges in Cloud performances (Xin Li, 2014)

As per (Xin Li, et al., 2014) in cloud computing (figure 1.5) the major issues are cloud security (74.6%),cloud performance (63.1%),availability (63.1%), hard to integrate with in house IT (61.1%), not enough ability to customization (55.8%), cost (50.4%), bringing back in-house (50.0%), regulatory requirements prohibit cloud (49.2 %) and not enough major suppliers (44.3%).

To detect the load fluctuations earlier will cause forecasting the destiny load conduct among interconnected tasks. Moreover, analyzing load variations may be beneficial in terms of figuring out the anomalies and threats in workflow programs that could lead to effective decision making.

1.6 THESIS ORGANIZATION

The main objective of this research work is to improve the cloud performance by efficient management of various performance improvement parameters the layout of the dissertation is as follows.

This complete thesis is organized into the seven chapters. Chapter 1 covers introduction work, describes problem statements and objective of the work. Chapter 2 discusses the literature review and provides a comprehensive review of load balancing concepts. It reveals the benefits and challenges of the existing methods and reflects the possible solutions for applying more optimized load balancing techniques in cloud-based systems.

Chapter 3 discusses the proposed performance improvement model and research methodology. Chapter 4 discusses Phase-I of the proposed work, Modified fuzzy Logic and Advanced Particle Swarm Optimization Model for Cloud Computing (MFL-APSO). Chapter 5 discusses Phase-II of the proposed AAP-IMC performance model. The second phase is based on Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing (ADRS-DDMC). Chapter 6 discusses the final phase of the research, Advanced Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC). In the last Chapter 7 discusses conclusions and future work of the complete work.

CHAPTER 2

RELATED WORK

In this chapter, described various existing cloud performance improvement methods suggested by different researchers. The performance of the cloud computing system depends on various factors such as task scheduling, load balancing and optimum resource utilization.

2.1 REVIEW OF CLOUD COMPUTING

A new timesharing computer technology for cloud computing was introduced in 1961 by cloud researcher McCarthy. As an expert computer scientist, he was the first one who predicted that time sharing would lead to a more powerful computing model. He stated that in future computing power will be consumed as a public utility just like water and electricity (McCarthy 1970). His idea became popular in that time but gradually faded away in 1990. It was again at the beginning of 20th century that McCarthy's idea resurfaced in the new form that is called **cloud computing** today (Deepak Pool A. et al., 2014).

Since 1970, when mainframes computers were introduced to IT industry, computing generations have gone through dramatic changes by several new innovations and ideas (Chun-Wei Tsai, et al., 2013). In the year 1980, during the recessionary phase, of IT industry, personal computers appeared to increase the efficiency levels of the businesses and individual users by increasing their profitability in that period. In 1990, client-server architecture offered new capabilities such as LANs to enhance the user's productivity, using the model of the shared network.

Cloud computing evolved from generations of grid and middleware computing technologies. Cloud computing paradigm provides scalable and virtualized resources. Virtualization is useful for resource sharing and optimum utilization of resources. Since users are not maintaining the IT resources, hence they are free to concentrate on business rather than on IT, that is looked after by specialized and trained manpower of cloud provider. Figure 2.1 shows the complete review of cloud computing. It has three major services IaaS, PaaS and SaaS (as discussed in section 1.1.4), four service models, private cloud, public cloud, community cloud and hybrid cloud model (see section 1.1.3, for more details).

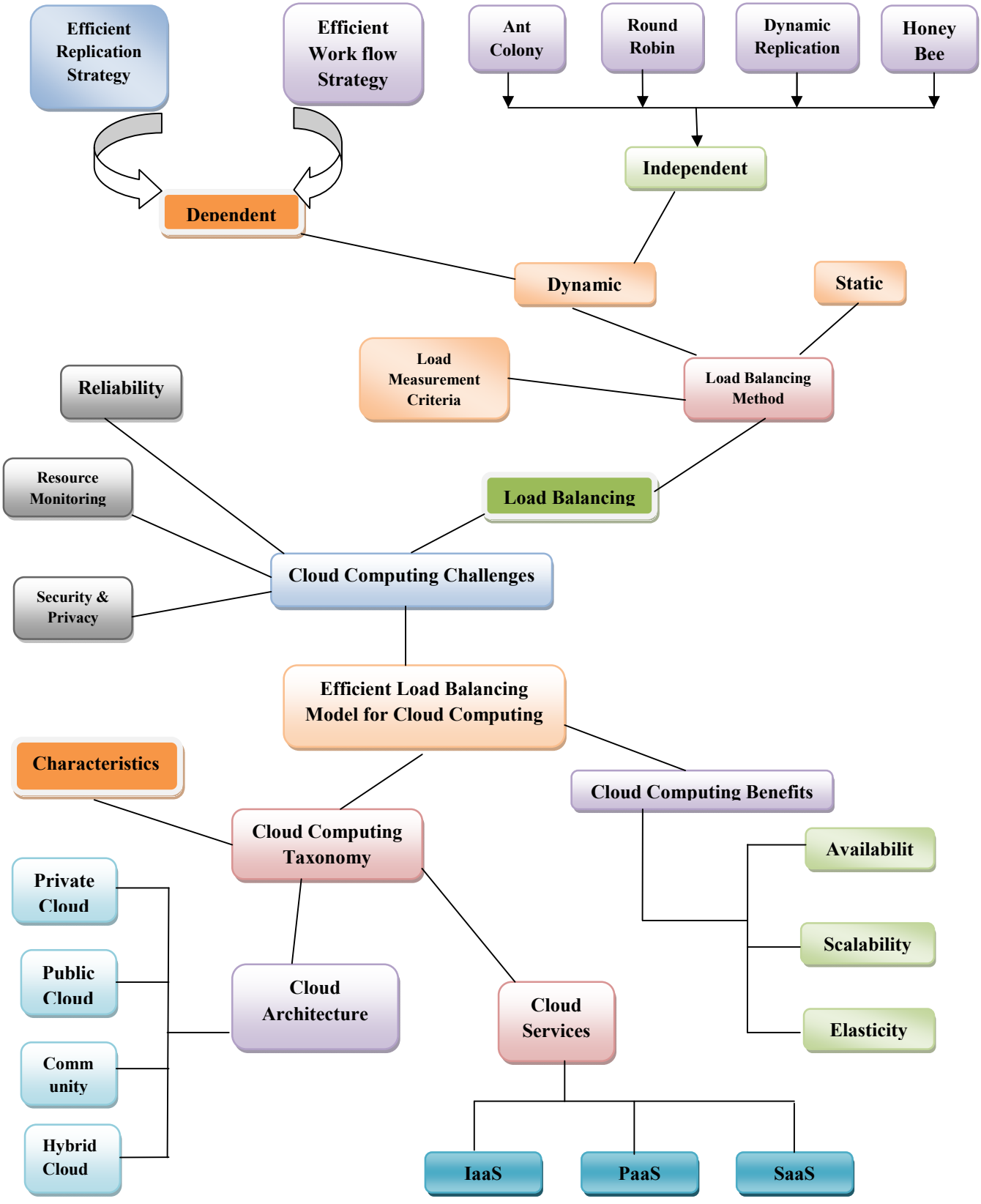

Figure 2.1 Review of Cloud Computing

2.2 RELATED WORKS FOR PERFORMANCE IMPROVEMENT IN CLOUD

Following research papers are related to the research in cloud performance:

Abbadi I.M., et al., 2014: Investigated towards trustworthy resource scheduling in clouds computing. In this work Abbadi I.M, et al., suggested an efficient scheduling model. Abbadi I.M, et al., focused on improving the performance of cloud by using some of the load balancing algorithms. A Load balancing method distributes the workload among different data center or load balancing to minimize the waiting time and to reduce the energy consumption in the data center. As per Abbadi I.M., et al., to improve the performance in cloud computing some of the useful work has been carried out in the past and suggested the improvement measures either at the operating system level or as an add-on application. Measures implemented at operating system are known as load balancing and improves the performance by using the load balancing algorithm.

Mohan N. R. Raj, et al., 2012: Investigated the resource allocation techniques in cloud computing research challenges for applications. According to Mohan N. R., et al., in software-based balancing methods the entire load is shifted from one data center to another considering the availability of the servers. Task load balancing for better the performance of cloud computing systems includes a large set of parameters such as workload, system characteristics and management policies proposed a scalable stochastic analytical model capable of quantifying the performance of IaaS. To minimize the interactivity the overall solution is obtained by dividing into sub-model solution iteratively. The solution obtained compared with the sub model to determine the efficiency of the proposed model.

Arunya M.R, et al., 2014: Investigated an adaptive fault tolerance management in cloud computing using replication and fragmentation. According to Arunya, et al., cloud performance can be improved by efficient resource management. Comparison and numerical analysis of various load balancing algorithms by considering the blocking probability of on-demand request. This research work described all the possibilities of energy saving that can be achieved in operational range by allocating the virtual machine based on priority. To evaluate the cloud performance considering the diversity of user requests and nature of cloud data centers proposed

an approximate analytical model for cloud servers. This model is based on establishing the relationship between a number of servers and buffer size that is required for the optimum performance. During this usage, the study of performance indicator such as blocking probability, mean number of tasks and probability that task will be served immediately, is carried out.

Jadeja Y. K., et al., 2012: Investigated the cloud computing concepts, architecture and challenges, to provide affordability and to improve the performance of cloud storage. An integrated overlay network of the storage cloud resources is placed in CDN in a manner so that it provides a low-cost, high-performance solution. Meta-CDN capable of reducing the multiple storages by intelligently matching and placing the needed users content at one or more storage provided considering their quality, performance and budget preferences. To address the file duplication and to improve the performance of the storage cloud, a new data management structure proposed i.e. Index Name Server (INS).An INS was capable of data de-duplication with nodes. It manages and optimizes as per the client side transmission conditions.

Jing Xiao, et al., 2012: Investigated the fault-tolerant and reliable computation in cloud computing. To avoid failure in web 2.0, applications composed of several components and need to be available to end users throughout their execution lifecycle proposed a solution to find out the optimal number of component types needed on each node. Furthermore, these nodes cannot exceed a maximum threshold value and the total running costs of the applications need to be minimized for that a suboptimal solution is also given. These above solutions are totally relying on cloud-based performance algorithms to satisfy their goals. The efficiency of the suboptimal algorithm is studied with respect to its success rate. To minimize the total end to end delay for a single input and to maximizes the total frame rate for streaming applications cloud researcher Jing introduced a new dynamic programming based optimal solution and prove the NP-completeness of the problems, for each of which a heuristic algorithm based on an optimization procedure was proposed.

Jisu Park, et al., 2012: Investigated an algorithm to improve the performance of IaaS based multiple cloud services and resource optimization with primitive task-based execution, which increases the total utilization of the cloud. This algorithm is capable of adjusting the computing

resource allocation dynamically, that is based on the actual task executions. To improve the performance of the access point researcher of this work suggested the methodology leveraging the web content, existing large-scale CDN infrastructures and the speed test network. This research work concluded by suggesting an ideal number of the data center be located in the different region to obtain minimum latency and throughput. To minimize the cost work Jisu Park, et al. has proposed a broker based on optimization algorithms to interface with multi-clouds. A cloud broker algorithm is mainly capable in decision making to choose the optimum cloud and distributing the service components to the different cloud or if needed migrating the components from one cloud to other in order to optimize the given criteria, so that the overall performance can be improved. The study is conducted both on static and dynamic environmental conditions of the cloud.

Junjie Ni, et al., 2012: Investigated the performance improvement of cloud computing. Considering the heterogeneity and proliferation of cloud ecosystem and proposed a broker-based approach that optimized virtual infrastructure need to be placed on multiple clouds and management of deployment in these clouds. In this work Junjie Ni, et al., described an overview of the current state of high-performance cloud computing technology, customer's requirements. Work advocated the use of virtual cluster from different cloud providers to provide the effective utilization of the resource and to avoid either underutilization or overloading. Considering the importance of quality of service Junjie et al., presented an approach to studying computer service performance in cloud computing. The work concludes the relationship among the maximum customers, minimal resources and the highest level of services.

Kun Li, et al., 2011: Investigated the multithreading method as a potential solution for performance improvement in a virtualized cloud environment. Where the combination of the different message is passed with multithreading since it improves the communication between resources and virtualized environment. Performance of the system is measured by taking a different load in two open sources software tools (Open Nebula and Eucalyptus). The workload is created from Wikipedia software and data. Physical location and the lazy allocation of the virtual machine image were the two key configuration choices that were made. Performance in AWS and GAE has been analyzed by simulation-based study considering scientific computing;

virtual good trading in social networks and social gaming to find out the dependency was carried out. The research work concluded by showing the performance dependency on the application. This would enable the user to select the cloud as per application.

Lee R. Bing Chiang Jeng, et al., 2011: Investigated the application of SOA in network virtualization and its effects are measured by comparisons of different methods. In this research work Lee R. Bing Chiang Jeng, et al., mainly described a new improved service-oriented architecture or framework for composing network and cloud services and modeled a new approach for composite network cloud services provisioning systems. To improve the performance Lee R. Bing Chiang Jeng, et al., focused on VM migration as a workload consolidation. This research work further studied the number of CPU and VM, the number of Virtual-CPU etc. to be shared for the optimum performance. An architecture consisting of front-end load balancer to route the load to the virtual machine and scaling algorithm for better utilization of resources introduced by Lee.

Liyang Xie, et al., 2011: Investigated the performance evaluation issues in SaaS. Liyang Xie, et al., proposed new models and improved metrics to evaluate cloud SaaS performance. Research work analyzed an evaluation approach based on Amazon EC-2 cloud technology. Multimedia-based applications need huge resources at the time of research and development. Whenever new algorithms are designed, its testing can be conducted on simulators. The test required a lot of computing resources and it takes either hours or number of day's incompletions that depend upon the size. Therefore, resource utilization is quite low, to address this issue this work proposed a method based on cloud paradigm where additional resources can be provisioned on demand and paying during the need. Results demonstrated that proposed scheme is cost-effective in comparison to the traditionally based system. Hence, testing requires less time, almost no investment in hardware can be very useful in accelerating the new developing activities.

Jenn Wei Lin, et al., 2013: Investigated load balancing in cloud computing environment using improved weighted round robin algorithm for non-preemptive dependent tasks. According to Jenn, et al., cloud computing uses the concepts of scheduling and load balancing to migrate or transfer tasks to underutilized VMs for effectively sharing the resources. The scheduling of the

non-preemptive tasks in the cloud computing environment is an irrecoverable restraint and hence it has to be assigned to the most appropriate VMs at the initial placement itself. Practically all the arrived jobs consist of multiple interdependent tasks and they may execute the independent tasks in multiple VMs or in the same VMs multiple cores. Also, the jobs arrive during the run-time of the server at varying random intervals under various load conditions. The participating heterogeneous resources are managed by allocating the tasks to appropriate resources by static or dynamic scheduling to make the cloud computing more efficient and thus it improves the user satisfaction. The main objective of this work was to introduce and evaluate the proposed scheduling and load balancing algorithm by considering the capabilities of each virtual machine (VM) the task length of each requested job and the interdependency of multiple tasks.

Kundu, et al., 2014: Investigated load balancing with tasks subtraction. According to Kundu, et al., cloud computing has been becoming popular day by day to provides a different type of web services and web resources and web applications to the web system. Cloud computing is working with web resources to execute applications smoothly. The objectives of cloud computing are to share network resources and web services over the Internet of web nodes. In cloud computing, load balancing is one of the target issues. The load is a measure of a number of works that a computation system performs which can be further classified as CPU load and network load. In this research work, the author proposed a new scheduling algorithm in a distributed system that chooses Suitable nodes with their subtracting tasks. This proposed approach provides efficient utilization of computing resources and maintains the load balancing in cloud computing environment.

Malik, et al., 2012: Investigated an optimal load balancing method for cloud computing. This method is based on efficient utilization of virtual machines. Load balancing method has an important and major concern in the cloud computing environment. Cloud computing technology comprises hardware, software and various resources. Managing these resources will play an important role in executing jobs from a remote location based request of cloud user. Now, these days' clients from various parts of the world are requesting or demanding for the various services at a rapid rate. In this present scenario, the load balancing methods should be very efficient in allocating the user request and also ensuring the usage of the cloud resources in an efficient and intelligent manner so that underutilization of the resources will not occur in the cloud

environment. In this research work, the author presented a novel virtual machine assign load balance method, which allocates all the incoming user requests to all the available virtual machines in an efficient manner. After that, the performance of the method was analyzed by using simulator Cloud-Sim and various results are calculated and compared with existing active virtual machine based load balance algorithm.

Punit Gupta, et al., 2012: Investigated a novel method for load balancing in cloud data center. In a large, scale cloud computing environment the end users and cloud data centers both are distributed geographically based across the globe. The most challenging task for a cloud data centers is to, how to handle and manages various services of the millions of requests, which are arriving very frequently from cloud users efficiently and correctly. In this research, the research work Punit Gupta, et al., described that in cloud computing load balancing method is requires to distribute the cloud workload dynamically equally in between all the cloud nodes. Efficient load balancing techniques help cloud service provider and user to achieve a high-level user satisfaction and optimal resource utilization and ensuring an efficient and fair allocation of every computing resource. In this research work Punit Gupta, et al., proposed new concepts, central load balancer based load balancing algorithm. The experimental results clearly show that proposed algorithm achieved much better load balancing into a wide scale area cloud computing environment as compared to previously existing load balancing algorithms.

Tharam Dillon, et al., 2010: an Investigated survey on scheduling and load balancing methods in cloud computing environment. Tharam Dillon, et al., described, now these days cloud computing technology is the most innovative and emerging method due to its some unique features and various qualities such as elasticity of resource provisioning and the pay per use-based pricing model, this model enables cloud users to pay only as per their need. Cloud services the user can able to access anywhere and anytime through required commodity hardware only, cloud service demands are increasing day by day. Cloud services must be providing a higher performance output to the cloud user and beneficial for the CSP. To achieving this important goal, many challenges are to be faced. Efficient load balancing is one of them, which helps the cloud service provider to meet the various qualities of service requirements of the cloud users and same times maximize his profit by efficient optimum use of the various cloud resources. For

balancing the total load into the cloud environments, the workloads and resources should be properly arranged and schedule in an efficient manner.

Rohit O. Gupta, et al., 2014: Investigated load balancers are used to identify which back-end servers are overloaded, for various balancing and scheduling algorithms. The selected cloud server allocates various IT resources and arranges or schedules all these applications dynamically on various free virtual machines, which are located on the same physical machine. It is also the responsibility of the cloud service provider dynamically allocate the various virtual machines across physical machines, for equal and efficient workload distribution and to avoid situation for all types of overutilization or underutilization of any cloud resources. In this research work cloud researcher Rohit O. Gupta, et al. described a new method to solve the problem of load balancing and task scheduling in cloud computing environments and described some of their shortcomings for further development. This work also described virtual machines migration issues involved related to load balancing.

Wu. H. Tantawi, et al., 2011: Investigated on honey bee behavior inspired load balancing of tasks in cloud computing environments. According to Wu. H. Tantawi, et al., scheduling of tasks in cloud computing is a P-hard optimization problem. Load balancing of non-preemptive independent tasks on virtual machines (VMs) is an important aspect of task scheduling in clouds. Whenever certain VMs are overloaded and remaining VMs are under loaded with tasks for processing, the load has to be balanced to achieve optimal machine utilization. In this research work Wu. H. Tantawi, et al., proposed an algorithm named honey bee behavior inspired load balancing (HBB-LB), which aims to achieve well-balanced load across virtual machines for maximizing the throughput.

Wu. H. Tantawi, et al., not only balances the load but also takes into consideration the priorities of tasks that have been removed from heavily loaded virtual machines. The tasks removed from these VMs are treated as honey bees, which are the information updates globally. This algorithm also considers the priorities of the tasks. Honeybee behavior inspired load balancing improves the overall throughput of processing and priority based balancing focuses on reducing the amount of time for the task has to wait in a queue of the VM. Thus, it reduces the response of time of VMs.

Tiwari, et al., 2010: Investigated a cost-effective load balancing based on honey bee behavior in a cloud environment. According to Tiwari, et al., in cloud computing environment, the load balancing of an emotive independent task is an important aspect of task scheduling. The tasks are executed on VMs and these VMs are run in parallel so that the load has to be well balanced across all VMs. Load balancers are used to identify which back-end servers are overloaded, for various balancing and scheduling algorithms. It is also the responsibility of the cloud service provider dynamically allocate the various virtual machines across physical machines, for equal and efficient workload distribution and to avoid situation for all types of overutilization or underutilization of any cloud resources. In this research work Tiwari, et al. described a new method to solve the problem of load balancing and task scheduling in cloud computing environments and described some of their shortcomings for further future development. Tiwari, et al., also described virtual machines migration issues involved in load balancing.

Amit Kumar Das, et al., 2013: Investigated survey on load balancing techniques and various challenges are addressed for cloud computing. Cloud computing is a new emerging technology, which has provides a new standard for large-scale parallel computing and distributed computing. Cloud computing provides sharing of resources, information, various software packages and other computing resources as per client requirements at the particular time. Cloud computing is growing continuously and rapidly. More cloud users that are newly attracted towards various utility computing, better and fast service. For better management and utilization of available cloud resources, more efficient methods for load balancing are still required. Therefore, that study of loads balancing methods in cloud computing environments is an interesting area of research for researchers. In addition, by efficient and better load balancing strategy in the cloud, increased performance of cloud environments and the user gets better services. In this research work Amit Kumar Das, et al., presented and discussed different load balancing techniques used to solve the issue in cloud computing environment.

Bakhtin Meroufel, et al., 2013: Investigated a new VM load balancing algorithm and then implemented in cloud computing environment using Cloud-Sim toolkit, in Java language. In this algorithm, VM assigns a varying amount of the available processing power to the individual application services. These VMs have different computing capacity, processing powers, the tasks or requests (application services) are assigned or allocated to the most powerful VM and then to

the lowest and so on. Bakhtin Meroufel, et al., have optimized the given performance parameters such as response time and data processing time, giving an efficient VM load balancing algorithm i.e. weighted active load balancing algorithm in the cloud computing environment.

Faragardi, et al., 2013: Investigated execution and performance analysis of various load balancing methods for cloud computing environment. As per Faragardi, et al., the idea behind the concept of cloud computing method has also significantly changed the field of different computing methods. For examples parallel computing and distributed computing systems. Cloud computing technique allows and enables a huge number of cloud users to access different type of distributed, hardware, software scalable and virtualized infrastructures over the Internet. Load balancing is a method equally distributes the workload of various nodes in between different multiple computers or on other computing resources over the computer network links to achieve optimal resource utilization, maximize system throughput, minimize response time and avoid overload. With recent innovation of technology and use of, resource control or load balancing in cloud computing is a main challenging issue. The main objective of this research work was to, find out identical qualitative components for various simulations in a cloud environment. Faragardi, et al., also described execution analysis of load balancing algorithms based on these components.

Jhawar R., et al., 2012: Investigated a novel dynamic round robin method for load balancing in a cloud computing environment. Most of the existing loads balancing methods working in cloud computing are carried out under homogeneous resources. However, today's requirement has been diversifying with the ever-increasing heterogeneity of computing resources in the cloud. In this research work Jhawar R., et al., presented the effect of round-robin method with dynamic behavior for cloud computing. By changing the important parameters of cloudlet long length, VM image size, host bandwidth and VM bandwidth. Cloud load balancing methods can improve by setting new dynamic round robin method parameters. Implementation of the proposed method was done on Cloud-Sim simulator for this implementation.

Liang Luo, et al., 2012: Investigated a new load balancing model. The proposed model is based on new improve partitioning method for the public cloud improve partitioning. The load balancing method aims to the public cloud, which has few numerous nodes with new distributed

computing in for various locations. This proposed model equally divides a public cloud into too many cloud partitions. When the cloud computing environment is so big and based on multifaceted, these divisions more simplify the load balancing. There are many efficient loads balancing methods are available like as the weighted round robin method, random algorithm and the new dynamic round robin method. When the cloud computing partitioning based methods are standard, cloud jobs comes faster as compared to its idle state and the situation is composite, as a result, a new strategy is used for the load balancing. Every user demands that his tasks should be executed in the shortest time, which creates a reason public cloud; more requires a new method that can help to complete the jobs of all cloud users with efficient reasonable response time.

Randles, et al., 2014: Investigated a new load balancing method for cloud computing. Randles, et al., described various cloud computing requirements such as efficient and improved access control, data availability, migration, security, various trust issues and sensitive information. This information might be more useful in the research associated with cloud computing method, which is used in the cloud computing environments for balancing of various loads. The concept of genetic programming permits the evolution of computer programs which can able to perform alternative computations conditioned on the outcome of intermediate evaluations. That executes more computations on variables of different types that can able to run on various iterations and recursions to achieve the expected outcomes. That subsequently defines an advanced distributed load balancing method. It is still an active area of research for the researcher to identify and an efficient load balancer which can improve the performance of a distributed system.

Tchana, et al., 2012: Investigated an optimized cloud partitioning method for load balancing in cloud computing. Optimized cloud partitioning method firstly partitions or divides the cloud environment into various partitions by efficient use of cloud-based methods such as clustering method, helps the cloud service providers to more simplify the process of load balancing. This proposed technique achieves higher performance and more stability for cloud computing environment. In the cloud, incoming patterns of jobs are often changeable in nature; therefore, resource allocation and job processing of multiple user requests over cloud environment, among various nodes, is a complex problem. The competence of each cloud node also diverges from each other.

Vilutis G., et al., 2012: Investigated a new load balancing concept called **Ananta load balancing** method for cloud computing. **Ananta means infinite in Sanskrit**. Ananta load balancing method examines the basic requirements for efficient load balancing. Ananta method is a scalable software load balancer and NAT that is optimized for multitenant clouds. It achieves scale, reliability and any service anywhere via a novel division of the data plane functionality into three separate tiers. At the second tier, a scalable set of dedicated servers for load balancing called multiplexers (Mux) maintain connection flow state in memory and does layer four load distributions to application servers. This design enables greater than 80% of the load balanced traffic to bypass the load balancer and go direct, thereby eliminating throughput bottleneck and reducing latency. This division of data plane scales naturally with the size of the network and introduces minimal bottlenecks along the path.

Souza. G.F.M., et al., 2003: Investigated load balancing model and presented new system architecture for cloud users. The main objective of the proposed method is to make resource requests in a cost-effective manner and discussed a scheduling scheme that provides good performance and fairness simultaneously in a heterogeneous cluster. For achieving this proposed method adopted as a shared metric. By considering various configurations possible in a heterogeneous environment, we could cut the cost of maintaining such a cluster by 28%. In addition Souza. G.F.M., et al., also proposed a scheduling algorithm that provides good performance and fairness simultaneously in a heterogeneous cluster. By adopting progress share as a shared metric and able to improve the performance of a job that can utilize CPUs by 30% while ensuring fairness among multiple jobs.

Sheheryar Malik, et al., 2011: Investigated a modified load balancing scheme. The cloud users of cloud services pay only for the number of resources (a pay-as-use model) used by them. This model is quite different from earlier infrastructure models where enterprises would invest huge amounts of money in building their own computing infrastructure. Typically, traditional data centers are provisioned to meet the peak demand, which results in wastage of resources during non-peak periods. To alleviate the above problem, modern-day data centers are shifting to the cloud. The important characteristics of cloud-based data centers are making resources available on demand. This enables the pay as per use model, that is, cloud users pay only for the services

used and hence do not need to be locked into long-term commitments. As a result, a cloud-based solution is an attractive provisioning alternative to exploiting the computing as a service model.

Punit Gupta, et al., 2013: Investigated load balancing of nodes in the cloud using Ant colony optimization. This is a modified approach to Ant colony optimization that has been applied from the perspective of cloud or grid network systems with the main aim of load balancing of nodes. This modified algorithm has an edge over the original approach in which each Ant build their individual result set and it is later on built into a complete solution. The approach aims at efficiently distribution of the load among the nodes and such that the Ants never encounter a dead end for movements to nodes for building an optimum solution set. This is a modified approach to Ant colony optimization that has been applied from the perspective of cloud or grid network systems with the main aim of load balancing of nodes. The main benefit of this approach lies in its detections of overloaded and underloaded nodes and thereby performing operations based on the identified nodes. This simplistic approach elegantly performs our task of identification of nodes by the Ants and tracing its path consequently in search of different types of nodes.

Xin Lu, et al., 2012: Investigated performance parameter of virtual machines in real time. The overloaded is easily detected once these parameters (such as capacity, processing power) exceeded the threshold. Quickly finding the nearest idle node by the Ant colony algorithm from the resources and starting the virtual machine can bear part of the load and meets these performance and resource requirements of the load. This realizes the load adaptive dynamic resource scheduling in the cloud services platform and achieves the goal of load balancing.

Radojevic, et al., 2011: Investigated a novel grid scheduling heuristic that adaptively and dynamically schedules task without requiring any prior information on the workload of incoming tasks. This model converts the system in the form of a state transition diagram with job replication to optimally schedule jobs. This algorithm uses prediction information on processor utilization. In this algorithm Radojevic, et al., uses the concept of job replication. In job replication method a job can be replicated to another resource if that resource completes execution of the current job and it is currently allocated to the job. This algorithm uses two types of queue namely, waiting for queue and execution queue. This approach is based on exploiting

information on the processing capability of individual grid resources and applying replication on tasks assigned to the slowest processors. The approach facilitates replication of tasks and assigned to execute on slower machines, on machines with higher processing capacity. In this approach, the communication costs are ignored. The experimental result clearly shows the better performance of this approach compared to traditional round robin algorithm.

Wang Juaan, et al., 2010: Investigated the task scheduling of the cloud storage system. Firstly, Wang Juaan, et al., analyzed the differences within the cloud storage, data grid and cloud computing. Also, point out that existing PSO based task scheduling algorithm of cloud computing is lack cost control ability and cannot ensure the solution is in the exits solution space which has to be guaranteed in cloud storage system. In order to address these problems Wang Juaan, et al. improved existing PSO algorithm by defining a cost vector and limiting the initialization solution and the solution search space in the existing solution space. The simulation results show IPSO algorithm can obviously save the execution time and offer meaningful solutions for cloud storage system.

Shi-Chen, et al., 2012: Investigated a novel approach to load balancing in cloud computing by using mathematical statistics. This algorithm considers the priority of jobs for scheduling and named as priority-based load balancing algorithm. It is based on multiple criteria decision-making model. A pairwise comparison based on multiple criteria and multiple attributes method was first developed by Thomas Saaty in 1980 and named as Analytical Hierarchy Process (AHP). Consistent Comparison Matrix (CCM) is the foundation of AHP, so to use the concept of AHP comparison matrices are computed according to the attributes and criteria's accessibilities. In this algorithm, each job requests a resource with determined priority. So a comparison matrix of each job according to resources accessibilities is computed and also comparison matrix of resources is computed. For each of the comparison matrices priority vectors (vector of weights) are computed and finally, a normal matrix of all jobs is computed named as new algorithm M. Likewise, another normal matrix γ of all resources is also computed. The next step of the algorithm is to compute priority vector of S (PVS), where S is set of jobs. PVS is calculated by multiplying matrix M with matrix γ. The final step of the algorithm is to choose the job with maximum calculated priority, so a Suitable resource is allocated to that job. The list of jobs is updated and the scheduling process continues till all the jobs are scheduled to a Suitable

resource. Experimental results indicate that the algorithm has reasonable complexity. Also, there are several issues related to this algorithm such as complexity, consistency and finish time. Then the performance of the system can improve and represent by-

a) **Efficient Traffic Management (T)**

   Low, Medium, High

b) **Efficient Disk Storage (M)**

   Small, Medium, Large

c) **Effective CPU Utilization (CPU)**

   Low, Medium, High

2.2.1 Existing Load Balancing Methods (LBM)-In general, load balancing algorithms are categorized (Moreno diano, et al., 2011) into two main groups (figure 2.2.1.1) as follows

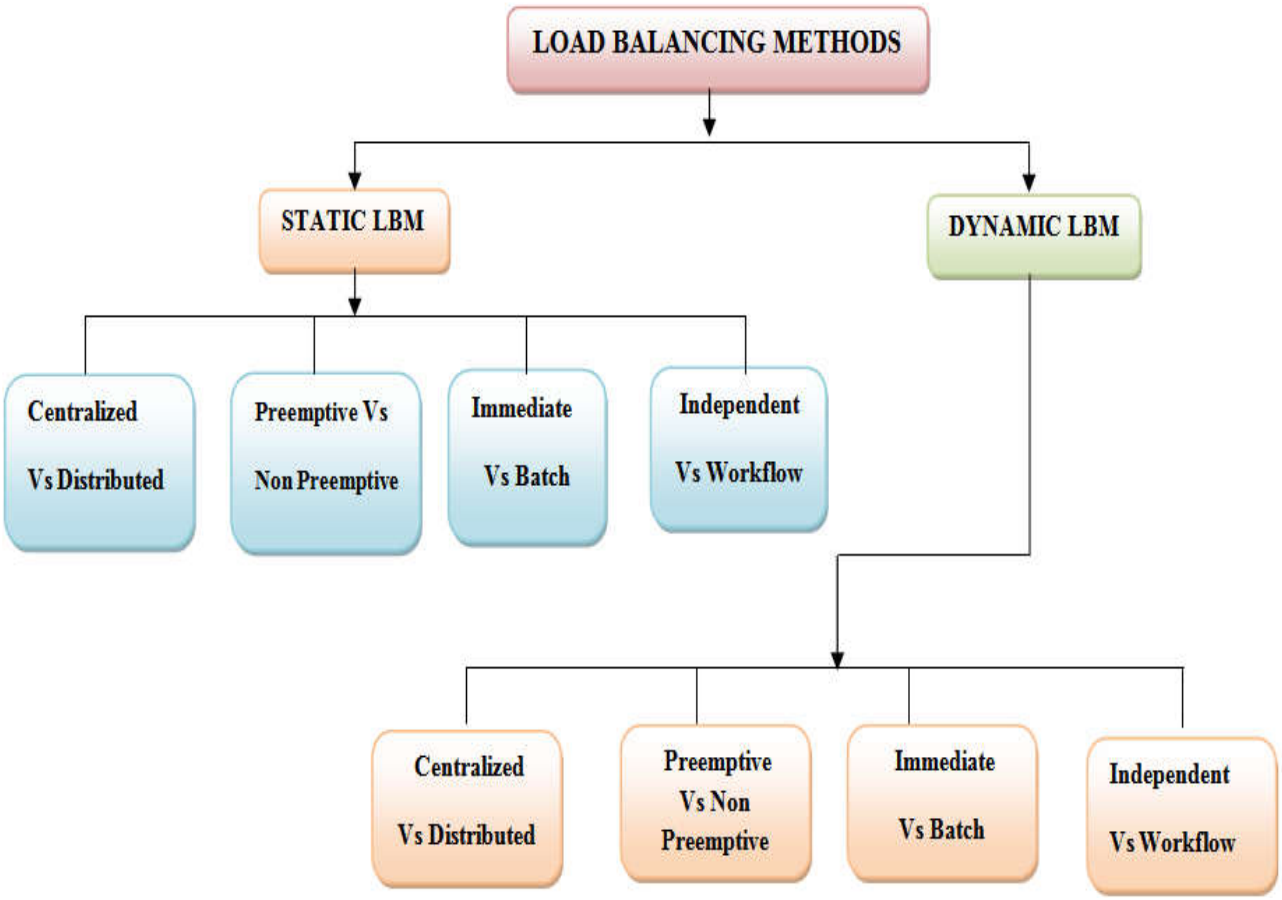

Figure 2.2.1.1 Types of Load Balancing Methods

A. Static Load Balancing Methods-In a static load balancing method, previous data regarding node capability, process power, memory and performance is required. The statistics needs cannot be modified at run-time. A general disadvantage of all static schemes is that these methods are based on pre-selection of the host for execution of a process and after selecting a host; the process can't change it (figure 2.2.1.2).

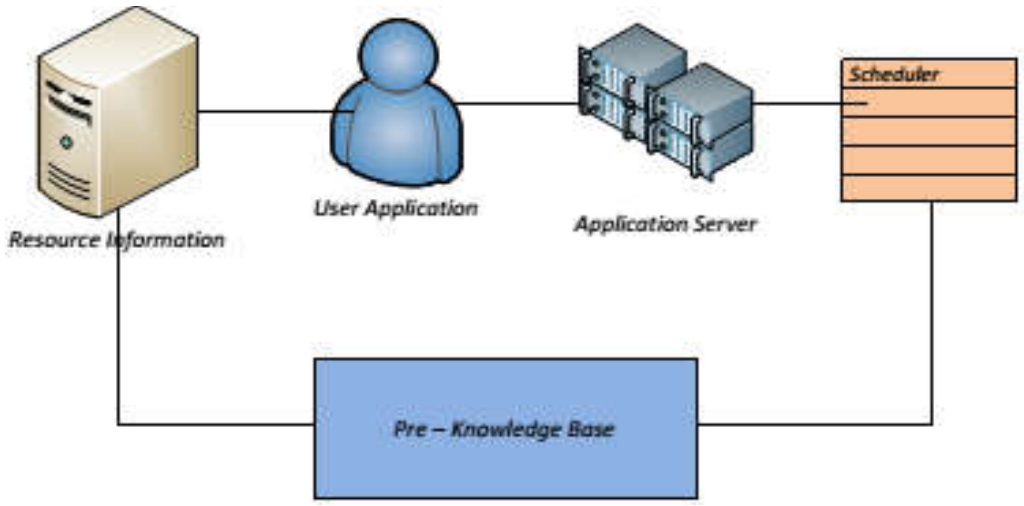

Figure 2.2.1.2 Static Load Balancing Method

B. Dynamic Load Balancing Methods-In this environment despite the need for prior, like static environment, the algorithms operate according to the run-time statistics (Nawsher Khan, et al., 2012).

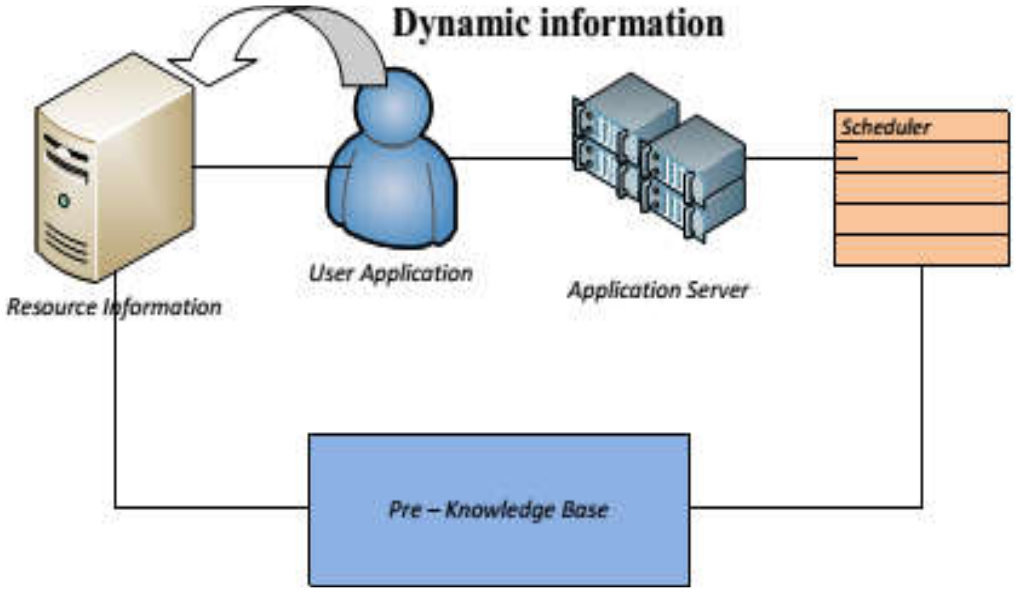

Figure 2.2.1.3 Dynamic Load Balancing Method

As shown in figure 2.2.1.3 the dynamic load balancing methods totally differ from the static load balancing algorithms in that the entire workload is equally distributed between the processors in advance at the runtime. Unlike a static method, dynamic methods are allocating processes dynamically and when any of the processors becomes underloaded or overloaded. Instead, they are buffered directly into the particular available queue, at on the host and it is also based on dynamical distribution.

Each of the static or dynamic algorithms could be divided into four different categories-

a) **Centralized Vs Distributed Method-**A centralized load balancing method is designed in such a way that a single unit name central controller. It can store all the required information for all the computing resources in a centralized manner. Centralized model is not so much popular and useful because it is not easily adaptable in terms of network scalability. Another load balancing method is distributed method. A distributed load balancing method does not require any centralized controller for storing computing resources. Over-centralized method, the distributed method performs outstandingly in terms of scalability and fault tolerance, so it can support elasticity.
b) **Preemptive Algorithm Vs Non-Preemptive Method-**A Preemptive load balancing method allows interruptions handling at the runtime. This method depends on the time factors. For example in the 'Queue', jobs are process based on their priorities. On the other hand, a non-preemptive method done not allows interruptions until all assigned tasks are scheduled on the available resources.
c) **Immediate Vs Batch Mode Method-**In immediate load balancing method, as soon as the jobs are assigned for the processing to the scheduler. It directly sent to the processing queue. In a batch processing method jobs are firstly grouped into the form of 'batches' and then jobs are processed batch-wise.
d) **Independent Vs Dependent Method-**In workflow load balancing the dependencies in between jobs should be calculated before it assigned to available computing resources. A dag figures and a Petri nets are the widely used languages to represent the various workflow load balancing. Independent modeling, however, will schedule the tasks without considering their interconnectivity (Das. P, et al., 2013).

2.3 EXISTING INDEPENDENT AND DEPENDENT METHODS

Independent and dependent load balancing methods can be categories in following types-

2.3.1 Dependent Load Balancing Methods-Therefore, to highlight the main gaps independent load balancing methods; we selected the following major workflow load balancing algorithms (table 2.3.1.1) and compared them as follows:

a) **Transaction Intensive Cost Constrained (TICC)-**Similar to market-oriented workflow algorithms, upon load balancing the tasks on available resources costing and timing will be considered in this algorithm. The overall goal of this algorithm is to minimize the execution time under the considered deadline (Rohit O. Gupta, et al., 2014).

b) **Heterogeneous Earliest Finish Time (HEFT)-**In this method, tasks will be ranked based on their execution time (Ravi Jhawar, et al., 2012). Tasks with lower execution time will gain the highest priority for resource load balancing.

c) **Scalable Heterogeneous Earliest Finish Time (SHEFT)-**It is considered as an extension to HEFT methodology described above. In this method, the earliest finish time and start time for each task will be calculated and from the calculated results, which task has lowest or minimum execution time will be allocated first to a computing resource.

d) **Revised Discrete Particle Swarm Optimization (RDPSO)-**This algorithm considers the discrete characteristic of PSO to optimize the make-span while minimizing the associated costs. In this algorithm, each particle will learn their best position locally and globally. The condition for each movement could be defined with different QoS elements such as deadline, budget or data transfer rate (Shi-Chen, et al., 2012).

e) **QoS Heuristic Workflow Load Balancing (QHWLB)-**The goal of this algorithm is on CPU utilization. In this algorithm, each task in a workflow will be analyzed according to their start time, finishing time, favorite processor and favorite predecessor.

f) **Dynamic Critical Path (DCP)-**This algorithm will determine the efficient mapping of the various jobs and instruction based on the dynamic critical path. In this algorithm, tasks will be prioritized on their estimated completion time. Scalability and fault tolerance is not considered in this algorithm (Sunil Kumar S. Manvi, et al., 2013).

g) **Deadline Constrained Workflow Load Balancing (DCWLB)-**This algorithm mainly is trying to minimize the execution time by meeting the deadlines based on QoS requirements.

In this algorithm, workflow tasks will be scheduled by following the concept of the partial critical path.

h) **Deadline and Budget Constraint (DBC)-**Most of the load balancing algorithms are trying to schedule the tasks on budget and time. But it is hard to satisfy these two requirements together.

i) **PSO Based Heuristic for Workflow Load Balancing (PSOHW)-**In this heuristic algorithm computation cost and data transmission cost should be considered. Each task has its own communication cost, which could be considered as a weight for each task. Using this methodology the workflow makespan will be minimized (Tao Chen R, et al., 2011).

j) **Genetic Algorithm (GA)-**Genetic algorithm (Broto L.D, et al., 2012) is one of the most revolutionary and promising algorithms for cloud computing which are inspired by evolutionary biology. In this algorithm, the solutions are shown with strings known as a chromosome.

Summary of Dependent Load Balancing Algorithms

Algorithm	Nature	Advantages	Limitations
Genetic Algorithm (Shi-Chen et al., 2012)	Dynamic	Minimizing the makespan and costs.	Fault tolerance is not considered.
Transaction-Intensive Cost Constrained (Qiang Guan et al., 2012)	Static	Minimize cost based on considered time.	Only will consider the in time task load balancing processes.
PSO (Liang Ma et al., 2013)	Dynamic	Makespan will be minimized.	Scalability and fault tolerance is not considered.
SHEFT (Hongyan Cui et al., 2013)	Dynamic	Optimize execution time.	Overhead will be added to pre-calculating the start/finish time for the tasks.
QoS Heuristic Workflow (Dodding Probhuling et al., 2013)	Static	CPU utilization is achieved by applying different parameters of QoS.	Parameters trustworthiness and fault tolerance are not considered in this algorithm.
Deadline and Budget Constraint (Han Xingye et. al 2010)	Static	Suitable for large scale distributed systems.	Do not consider communication and execution cost of the tasks.

Dynamic Critical Path (Junjie Ni, et al. 2011)	Dynamic	Tasks will be prioritized base on their completion time.	Do not consider scalability and fault tolerance.
HEFT (Janpet, J. et al., 2013)	Static	Improved Makespan.	Scalability and resource utilization are not considered within this algorithm.

Table 2.3.1.1 Summary of the Reviewed Dependent Load Balancing Algorithms

2.3.2 Independent Load Balancing Algorithms-This section covers the independent load balancing algorithms. Table 2.3.2.1 includes the benefits and challenges of the explored algorithms.

a) **Round Robin-**This algorithm is considered as a static load balancing algorithm. Round robin operates based on the time spans assigned to each node. With this algorithm, traffic will be distributed evenly. However, as the algorithm is static, it cannot manage the network's load in a real-time manner (Thanadech, et al., 2013).

b) **Dynamic Round Robin-**This algorithm is an improved model of round robin algorithm. In this model, the tasks execution sequence will be recorded automatically based on the current status of the network. The algorithm is creating less overhead than a Round-Robin algorithm and it can improve the response time. However, it has low performance in busy environments (Tin Yu, et al., 2012).

c) **Signature Patterning-**This algorithm works with different time-slots. It captures signatures from executed tasks and resources to make patterns. If the patterns show that the resource reached its threshold, based on the captured signatures, the load will be distributed on less overloaded nodes (Tiwari, et al., 2012).

d) **Task Consolidation Algorithm-**The algorithm (Katarina Stanoevska Slava, et al., 2010) will operate dynamically according to the heuristic methodologies. Different components will be considered for the task allocations.

e) **Dynamic Replication Algorithm-**In this method, the algorithm will try to replicate the files on local servers. In this case, when users want to access a file, they could find it on the local server. Using this method the access time would be minimized (Vilutis G., et al., 2012). Dynamic replication helps in efficient load balancing.

f) **Map Reduce-**This algorithm has two main functionalities (Wai Leong, et al., 2011)

- **Firstly,** all the jobs will be divided (mapped) into subtasks. Each subtask will have its own key ID. In next step, all the IDs would be converted to hash keys. Then, the reduce function will do an operational summary on each sub-groups with their hash IDs and it will generate a single output.
- **In the final stage,** a central node is dedicated to compare all the generated single outputs and assign them to available resources. The only problem with this algorithm is related to its overhead. As mapping and reduction step can be done in parallel; nodes might be overloaded.

g) Ant Colony-In this algorithm, Ants will move forward to find the first under loaded resource. If the Ant finds the first under-loaded server, it will move forward to check the next server status. If the next resource is underloaded as well, it will move forward to find the last under-loaded resource. Otherwise, the Ant will move backward from overloaded node to the previously available resource (Wei Ma, et. al., 2012).

h) Index Name Server-The algorithm will minimize the data replication and data redundancy. Based on the maximum transition time and bandwidth, the algorithm will do some calculations to find an optimum server for task load balancing. In this algorithm, the connection weight between server and nodes will be calculated to clarify whether the server can handle more nodes or not (Weiguang Shi, et al., 2010).

i) Min-Min-In this algorithm, the minimum completion time for each task will be calculated. Then the tasks with the overall minimum completion time will be selected. The same iteration will happen until all the tasks are allocated to available resources (Bakhtin Meroufel, et al., 2013).

j) Max-Min-In this algorithm, the minimum completion time for each task will be calculated. Then the task with maximum completion time will be assigned to the first available resource. The iteration will continue until all tasks are assigned to the available resources (Chen, Yizeng. Li, et al, 2013).

k) Artificial Bee Colony-This algorithm is designed according to the behavior of the honey bees. In this model three main components exist, employed honey bees, unemployed honey bees and food sources. Bees will start looking for the food source randomly and they will record the information related to the location of the source and its profits (Mohan N. R., et al., 2012)

l) **The Discrete Version of PSO (DPSO)-**PSO is a type of evolutionary algorithm that works base on the velocity and portion of the particles. Each particle has a local memory that memorizes its velocity and position. Additionally, the particle can learn from its adjusted velocity. In this case by each movement based on the velocity location, limited between (-1, 0, 1), the position of the particle will be updated. If the Vt =-1, the particle will learn the new position from its neighbor, if Vt =0, there will be no change in the position and if Vt =1, the particle will learn from its past movement (Anshul Rai, et al., 2012).

Reviewed Independent Load Balancing Algorithms

Algorithm	Nature	Advantages	Challenges
Round Robin (Thanadech, et al., 2013)	Static	Distribute the traffic evenly based on time slices.	The real-time load cannot be considered Longer waiting time.
Dynamic Round Robin (Tin Yu, et al., 2012)	Dynamic	➢ Minimize the waiting time. ➢ Minimize response time.	The performance of the algorithm is low.
Signature patterning (Tiwari, et al., 2012)	Dynamic	Resource status and allocation are managed precisely.	Extra overhead will be added due to pattern capturing and comparison.
Map Reduce (Wai-Leong Yeow, et al., 2011	Dynamic	This method is suitable for large distributed networks.	Due to parallel processing, nodes can be overloaded.
Ant Colony (Wei Ma, et al., 2012	Dynamic	It uses a meta-heuristic approach.	This method is ineffective resource utilization is needed.
Index Name Server (Weiguang Shi, et al., 2010).)	Dynamic	Minimize the waiting time and Improved plan for fault tolerance.	Extra overhead will be added to servers due to connectivity calculation.
Max-Min (Bakhtin Meroufel, et al., 2013)	Static	Minimize the waiting time.	Cannot support fault tolerance.
Artificial bee Colony (Mohan N. R. et al 2012))	Dynamic	Suitable for scalability.	System throughput cannot be fully optimized.
DPSO (Anshul Rai, et al., 2012).	Dynamic	Improved availability of the resources Suitable for elasticity and scalability purposes.	Fault tolerance can be supported Resources cannot be fully optimized.

Table 2.3.2.1 Summary of the Reviewed Independent LB Algorithm

2.4 WORKFLOW LOAD BALANCING

Table 2.4.1 summarizes the highlighted works that have been done for improving the load balancing in workflow load balancing models. Analyzing the pros and cons of the selected workflow load balancing algorithms, most of the algorithms were focusing on optimizing the performance of the system while completing the workflow execution. However, in workflow structure applications, interconnected dependencies exist between available tasks. Therefore, it is essential to consider the load flow fluctuations among the existing tasks. With the emergence of the new technology of cloud computing, soon most of the data-intensive and scientific applications will run on cloud-based systems (Punit Gupta, et al., 2013).

Pros and Cons of Workflow Load Balancing Algorithms

Author	Pros	Cons
Ravi Jhawar et al., 2014	Central Load Balancer provides efficient load balancing among virtual machines in cloud data center.	More bandwidth and more CPU consumptions.
Dhinesh Babu L.D. et al., 2013	Achieve well-balanced load across virtual machines and maximizing the throughput.	More waiting time and CPU Usage.
Subhadra Bose Shaw et al., 2014	Load balancing can be improved by VM migrations.	VM migration issues involved in load balancing.
Tushar Desai et al., 2013	Provides shared resources, information, software packages and other resources as per client requirements at the specific time.	High bandwidth usage, Did not consider the tasks interdependency.
Kashi Ventakesh, et al., 2010	Select the replicas based on network performance.	More CPU usage.
P.Varalakshmi et al., 2011	Pre-replicates the files and data with fewer costs.	Consumes-more bandwidth.
Wei Ma, Xiaoyong Li et al., 2012	Efficient job load balancing and resource allocation techniques.	More CPU usage.
Ajay Gulati et al., 2013	Applies message passing in task level/job level load balancing.	More CPU and memory usage in job level load balancing.
Thanadech, et al., 2009	Applies Meta CDN tool for managing the workflow task with minimum makespan.	The tool is not practical for high volume data.

Table 2.4.1 Summary of the Reviewed Workflow Load Balancing Algorithms

2.5 REPLICATION METHODOLOGY

Evaluating the pros and cons of the mentioned replication techniques, although the proposed methods have had a great impact on optimizing the load balancing in cloud-based systems; a more robust replication method is required.

Pros and Cons of Various Replication Methods

Author	Pros	Cons
Anton Beloglazov et al., 2005	➢ Applies simple bottom-up and aggregation bottom-up method. ➢ Makespan is minimized	➢ More CPU usage More bandwidth consumption.
Chenn Jung Huang et al., 2013	➢ Pre-replicas are chosen based on access histories Total execution time is minimized by 10%	➢ Did not consider the effective network usage more bandwidth consumption.
Gaucho Xu et al., 2013	➢ Minimized the number of the replicas by pre-locating them on local servers.	➢ More memory usage.
Junkie Ni et al., 2011	➢ Applies figure probability function along with fault tolerance capability.	➢ More numbers of created replicas were chosen only based on access histories.
S.Sindhu et al., 2011	➢ Applies different file access patterns ➢ Minimized access rate. ➢ Minimized the total bandwidth consumption.	➢ More memory usage is needed for storing the access patterns.
Tiwari, A. et al., 2010	➢ Greater weight is applied to the latest accessed file which will be selected as a target pre-replica.	➢ More memory and CPU usage did not consider the effective network usage.
Qiang Guan et al., 2012	➢ Select the replicas based on network performance.	➢ More CPU usage.
Yifeng Geng et al., 2011	➢ Pre-replicates the files with less estimated costs.	➢ More bandwidth consumptions.

Table 2.5.1 Summary of the Reviewed Replication Methods

The new method will be capable of predicting the future needs of the users so it can pre-replicate the files before the requests have been submitted for them.

CHAPTER 3

PROPOSED PERFORMANCE IMPROVEMENT MODEL AND RESEARCH METHODOLOGY

In this chapter, a new performance improvement model is introduced **Advanced Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC)**. The proposed AAP-IMC performance improvement model presents a set of solutions for load balancing in the Cloud.

3.1 PROPOSED PERFORMANCE IMPROVEMENT MODEL

In the cloud, the resource optimization is the most important process. Nowadays, many organizations use cloud storage. Many requests can occur at a time in a cloud which gives more problems in the cloud. The important issues are server crash, failover and outage. Some outages are lengthy and it may take more hours or days to solve the problem. Cloud computing is a set of several virtual machines that are sliced into virtual servers and placed at different geographical locations for providing services to customers.

The present method deals with the following research challenges in the cloud:

1. **Efficient load balancing-**To achieves efficient load balancing.
2. **Optimization of resources-**To achieve optimum utilization of computing resources.
3. **Identification of reliable VMs-**To find reliable VMs effectively for better performance.
4. **Better performance-**To achieve better performance of the entire system.
5. **To predict load earlier-**Proposed method works sender as well receiver both of the ends. So it helps to predict load earlier.

3.2 PHASES IN PROPOSED MODEL

The proposed performance improvement model **Advanced Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC)**.The proposed **AAP-IMC** performance improvement model presents a set of solutions for load balancing in the Cloud. The

proposed model uses a combined strategy of two proposed methods **MFL-APSO and ADRS-DDMC** in various phases.

3.2.1 Phase-I: First phase is based on **Modified fuzzy Logic and Advanced Particle Swarm Optimization Model (MFL-APSO)** to optimize the total execution time of tasks in the workflow applications. It is based on a heuristic algorithm that uses MFL-APSO. The key objective of applying the MFL-APSO method is to minimize the total tasks execution time by verifying the load fluctuations of the interconnected tasks. The model optimizes the load balancing method by minimizing the total execution time. In order to manage the allocated tasks in cloud architecture, there is an application scheduler designed to balance the workload between available resources.

The variance of the algorithm considers factors such as load variations (fluctuations) and optimization of the data retrieval time. The proposed model MFL-APSO is validated by applying five workflow structures with different data block sizes. The results are compared with HEFT algorithm and SHEFT. By analyzing the memory use processing duration and data access time, application schedules, maximize the resource utilization. The first phase is the initialization phase. In the first phase the efficient load, expected speed, response time and execution time of each VM is to be found and achieved:

1. Efficient load balancing by performance model.
2. Provides effective resource utilization.
3. In workflow application mapped parent and child task accurately.

3.2.1.1 Role of Fuzzy Controller and Rules in MFLM-In proposed MFLM; fuzzy controller plays an important role. The fuzzy controller works as a feedback system by continuous repeating various cycles to generate the desired output. To establish the fuzzy controller modeling for proposed Modified Fuzzy Logic Based Model, firstly we need to define the various inputs and output data variables. Data center management is progressed by the performance improvement function which is calculated among three factors Bandwidth (BW), Memory (MEM) and CPU Cycles (CPU). Proposed Modified Fuzzy logic model based on following rules:

1) **Rule-1 Approximate Reasoning-**Fuzzy set defined as corresponding to the linguistic values. We include reasoning as multi-conditional in the form given, If- then Rule, rule-1 through and a fact is.
2) **Rule-2 Fuzzy Implication Rule-**A fuzzy implication is defined as the method of the optimum result and it gives any of possible true values of any given fuzzy propositions. Here positive values define the total true value for the conditional proposition, which denoted by IF Then rules. This is called a classical implication of rule of, from a restricted domain to a full domain of real true values. In this rule creation, we interpret negation and disjunction, as a fuzzy union and fuzzy complement and then 'in' classical logic is employed.
3) **Rule-3 Relation R-**The fuzzy relation employed in reasoning is derived from the if-then rules in (2). For each rule in given (2), we create a new relation. By the fuzzy formula, for each all, are defined by a set. This set is the union of all the relations and rules which are defined in 1. Here in this research work, we are considering the disjunctive nature problems such as prediction of performance parameters for cloud computing.
4) **Rule-4 Fuzzy Proposition Rule-**In fuzzy logic propositions are measured in its defined ranges and it should be based on true values (above the threshold). It depends on the matter of degree and their type. Each of the fuzzy propositions is uttered by various element intervals. Here in this advanced fuzzy logic rule creations method for Phase-I of the experiments, we consider this proposed model as condition base and an unqualified proposition.
5) **Rule-5 Compositional rule inference-**In this rule variables are considered which take a set of values from predefined sets and various, respectively and it also assumes for all and these all variables are related by a function.

3.2.2 Phase-II: Second phase of the research is based on **Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing (ADRS-DDMC).** This phase introduces a novel dynamic data replication method that is functioning based on Anticipations to create the pre-replicas for future needs of the sites. This method uses anticipatory data replication for increasing the reliability and availability of cloud data. Pre-replication can increase the data availability and robustness of the cloud systems and hence requested jobs can be completed with minimum execution time and high network usage output.

In the second phase, the reliability and failure VM is identified and in the last phase, efficient resource optimization is accomplished.

A load balancing model is designed and followed out to reduce virtual machine load with the function of load balance server amongst servers according to their processor or I/O usage and to experience a zero downtime of virtual cars in the process. A cloud load balancing algorithm to compare and balance is likewise proposed that is based on trying out and reaches equilibrium faster. The proposed ADRS-DDMC method optimizes load balancing by increasing the data availability among the existing sites. The results are compared with LRU (Least recently used) and LFU (Least frequently used).

These algorithms assure that each physical server has plenty storage and achieved:

1. Introduces a novel Anticipatory data replication method that is functioning based on dynamic Anticipations to create the pre-replicas for future needs of the sites.
2. To determines dynamic workflows for various workload earlier.
3. Efficient task load balancing and load balancing in the cloud.
4. To improve the performance of cloud computing system by improving short comes of existing methods.

The resource optimization gives more efficient load balancing with efficient parameters of the first phase and second phases.

3.2.2.1 Improved Replication Methodology for Phase-II: In the second experiment of this study replication methodology has been applied to improve the load balancing in cloud computing. The aim is to pre-replicate the files with high access probability. Therefore whenever there is a need for accessing these files, they would be accessed locally.

The method is structured based on the following details:

1) **Step-1** Each site has its own replica manager. There is also a central replication manager which monitors the sites' access and data catalog. The central replication manager will store the files name and the numbers of the time that the file has been accessed.
2) **Step-2** When a site finds that it needs a data file which is not stored locally, it asks the central replication manager to replicate the file from the available sites. Then the central replication

manager will perform heuristic A* search algorithm in the data catalog to find out the site that has the target files. A* is a popular heuristic algorithm, that can find the shortest path in the tree structure with minimum cost. Then between available sites the one that has the minimum bandwidth will be selected:

Communication cost = Size of the replica / Bandwidth between servers ---- (3.2.1)

3) **Step-3** This phase explains the replacement procedure. After replicating the files, they should be transferred to the site that was requesting the file. If the site has enough memory the file can be added easily, otherwise replacement procedure should be initiated. In replacement phase, Most Recent Used (MRU) technique has been applied which emphasizes on removing the old files from the site. So if-

$$\mathbf{T_{i+1}\ (current\ time) - T_i\ (last\ access\ time) > Threshold} \quad \text{------- (3.2.2)}$$

Where T = time and Threshold = minimum required time

3.2.3 Final Phase (AAP-IMC)-The final phase of this research work investigates Advanced Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC). The proposed AAP-IMC performance improvement model presents a set of solutions for load balancing in the cloud. Proposed AAP-IMC model uses a combined strategy of two proposed methods MFL-APSO and ADRS-DDMC in various phases. Managing accessibility for application performance is a key challenge for QoS. Once the input has been accepted the user request proceeds to the load balancer where the proposed AAP-IMC, the algorithm would execute.

These three methods (Phase-I, Phase-II and final phase methods) are used in cloud computing environments at the time of data loading and data access. Existing algorithms are used to compare these three proposed methods for cloud load balancing. The proposed AAP-IMC, MFL-APSO and ADRS-DDMC algorithms give more efficient than the existing algorithms. By using efficient load balancing, reliability and efficient optimum usage of the resources can improve the performance of cloud computing system.

3.2.3.1 Parameters for Proposed AAP-IMC Model-Cloud load balancing server allocated the load at the time of increasing the several CPUs or memories for their resources to scale up with the increased demands (table 3.2.3.1). This service is primarily applied to commercial enterprise

demands. In the cloud, the load balancer is a host to monitor the load and allocate the load to VMs by using load balancing algorithms. The load balancer is used for two significant processes, one is primarily to boost the availability of cloud resources and the other is secondary to promote performance. In late years, many research works have been sought on various challenges in cloud computing.

Parameters of MFL-APSO

	Phase-I (MFL-APSO)	Phase- II (ADRS-DDMC)
Parameters for Proposed AAP-IMC Model	Response Time	Mean Job Execution Time
	Execution Time	Effective Network Usage
	CPU Load	Total Number of Replicas
	CPU Time	Soft error rates on memory %
	Memory Rate	Resource Availability
	Workflow Soft Error Rates	Time Consumption

Table 3.2.3.1 Proposed Algorithm Parameters for Performance

The load balancing in cloud computing will provide following:

1) Distributes workload across multiple VMs in a data center.
2) Prevents over or underutilization of resources and optimizes performance.
3) Provides optimum and efficient utilization of computing resources.
4) Data replication increases the availability of data.

In the cloud, the proposed load balancing algorithm comprises of the observing parameters-

1. Latency or Response Time-(Tanya. et al., 2013) says that the latency is the time taken to send a unit of data between two points in a network. Here the latency is calculated by the time required for one packet to travel from load balance server to a virtual machine. In cloud computing, the network latency is the total of all the computing delays of the links between the various sender, receiver hosts, the switches and routers in between. The latency period is calculated as-

$$\textbf{Latency-Rate } L\_R = \frac{\sum^{n}_{t=0} (C\_T_t - A\_T_t) - E\_T_t)}{N} \quad \text{------ (3.2.3.1)}$$

Where, C_T - Completion Time, A_T - Allocated Time, E_T - Execution Time

The latency takes place inside the transmission range, for every millisecond. The average latency is calculated as the number of processes in the load balancer.

$$L\_R_{avg}=\sum_{i=1}^{n}\frac{L\_R_{avg}\,1+ L\_R_{avg}\,2+\ldots\ldots\ldots.. L\_R_{avg}\,N}{N} \quad \text{-------} (3.2.3.2)$$

In the cloud, the request is placed in the queue. The data center manager gets the request and sends the request to all the VMs and the user data is replicated. VMs which will responses in the time limit are considered as reliable. The time limit is calculated based on the performance of the process in the VM. $\mathbf{T= F+ (N/ \sqrt{NP}) + (P / \sqrt{P})}$ ---------- (3.2.3.3)

Where-

- L_R_{avg} = Average latency time.
- T= Total response time requests receive an interactive response in a cloud application.
- F =Fixed interval of the time in the worst case round trip latency for an environment.
- P = Processing nodes are the time for the application to run on one processor.
- NP = Number of the processor in a virtual machine. The response time of the VM can be done within the time taken in the.

2. Execution Time-(Xen Xing., et al., 2014) Execution time can be estimated by evaluating the time required for completing the operation time of virtual machines in the cloud data center. That is the full measure of the time required by the virtual machine for executing instructions, that is the time of an executing program, hence it is generally much less than the entire execution time of the course of study generally much less than the total execution time of the program.

$$\textbf{Execution Time ET} = \sum_{i=0}^{N}(F_T - S_T) \quad \text{----------} (3.2.3.4)$$

Where, F_T - Finishing time of, $i^{th,}$ job, S_T - Starting time of i^{th} job

In VM, the CPU time per execution is a performance index that reads the average obtained CPU time in microsecond during each running of the domain execution per second. The execution per the second metric measures the counts of the domain from being scheduled to work on a physical CPU over the unit time duration.

$$\textbf{Mean Execution Rate} = \frac{\sum_{i=0}^{n}(F_{Ti} - S_{Ti})}{\sum_{i=0}^{n} J_{Si}} \quad \text{----------} (3.2.3.5)$$

Where- J_S- Job Size

3.3 PHASES IN PROPOSED (AAP-IMC) MODEL

The primary focus of this thesis is to improve the efficiency of cloud data center to process client data. As the main focus of this dissertation is on dependent and independent load balancing algorithms, to highlight the certain path of this study from reviewed dependent and independent load balancing techniques, we have selected workflow load balancing and replication methodology as the main targets of this research. Figure 3.3.1 show phases in the proposed model.

Proposed performance enchantment model is based on two phases:

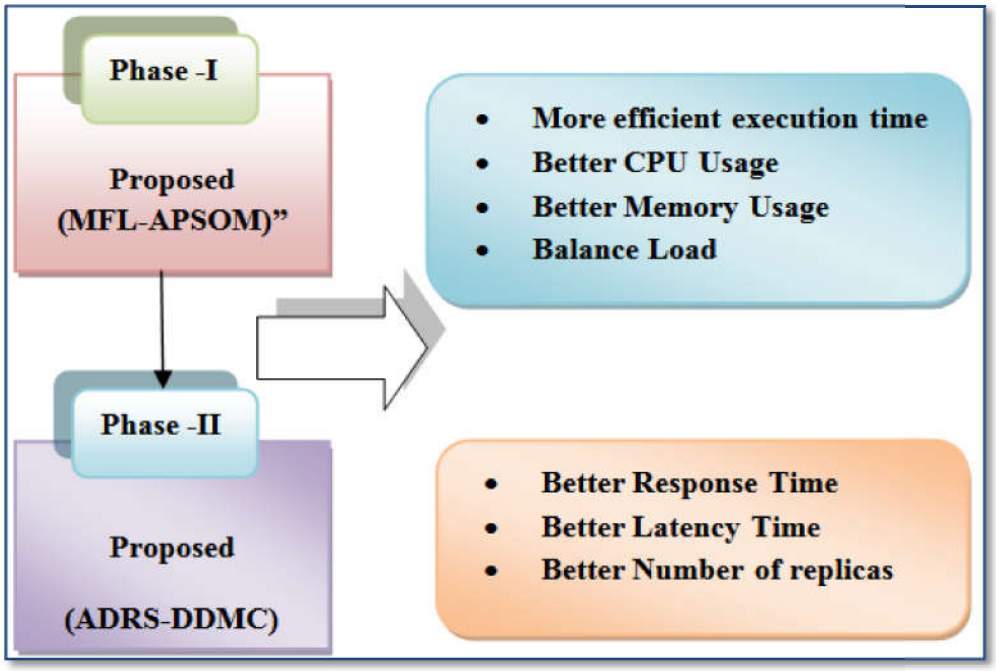

Figure 3.3.1 Phases in Proposed Performance Enchantment Model

1). Phase I-It is based on MFL-APSO. This first phase introduces novel fuzzy rules for workflow applications.

Expected Outcome for Phase I-The outcome of the experiment confirms the suitability of the optimization method and will achieve:

a) To optimize the total execution time of tasks in the workflow applications.

b) The key objective of applying the MFL-APSO method is to minimize the total tasks execution time by verifying the load fluctuations of the interconnected tasks.

c) The proposed model MFL-APSO is validated by applying five workflow structures with different data block sizes.

d) Make-span can be two times efficient as compared to existing HEFT and SHEFT Method.

e) Better CPU and Memory usage can be achieved than existing HEFT and SHEFT Method.

2). Phase II-This phase is based on ADRS-DDMC. Phase two of the work introduces a novel dynamic data replication method that is functioning based on anticipations to create the pre-replicas for future needs of the sites. The proposed ADRS-DDMC method optimizes load balancing by increasing the data availability among the existing sites. The results are compared with LRU and LFU.

Expected Outcome for Phase II-The method optimizes load balancing by increasing the data availability among the existing sites and will achieve:

a) Analyzing the algorithm's performance by

- **i.** Less response time
- **ii.** Minimum access latency
- **iii.** Efficient number of replicas
- **iv.** Less CPU time
- **v.** Optimum utilization of computing resources

3) Final Phase-This final phase proposed performance improvement model **AAP-IMC** for cloud computing. The proposed AAP-IMC performance improvement model presents a set of solutions for load balancing in the Cloud. The proposed model uses a combined strategy of two proposed methods MFL-APSO and ADRS-DDMC in various phases.

3.3.1 Advantages of the Proposed AAP-IMC Model-Some of the specific Advantages of the proposed methodology over the existing load balancing and reliability algorithm are given below:

- Increase the VM speed
- Reduce processing time
- Reduce context switching
- Reduce waiting time of the request in the queue
- Increase reliability by identifying failure VMs earlier
- Efficient resource allocation in VMs

3.4 RESEARCH DESIGN FOR PROPOSED PERFORMANCE MODEL

The quantitative approach is known as true science and it applies traditional mathematical and analytical approaches to analyze the results. In quantities research approaches, the hypothesis will be generated to be proved as a result of the experiments (Pannu, et al., 2012). Therefore, the approach has been selected as a research design for this study. Quantitative approach evaluates the existing challenges based on the available facts and the information gathered from the literature review. The hypothesis should be provable by mathematical and analytical methods which are generally shaping the experiments needed during the study. Additionally, quantitative research will construct the study in a manner that allows other users to repeat the experiments and generate the similar results (Liang Luo, et al., 2012).Research questions and aim of this study have been explained in chapter one. To validate the proposed model, two experimental scenarios and a research action study have been explored.

The first experiment adopts replication strategy to design a smart anticipatory replication algorithm that pre-replicates the files for addressing the future needs. The experiment employs heuristic approaches to optimize the load balancing in cloud computing through replication strategy. The second scenario uses limited servers available at the data center if the requests submitted are high than the capacity of the data center, its overall performance degrades for the entire system. In this situation an efficient load balancer is used in cloud computing, to improve the performance of data center.

3.5 RESEARCH METHOD

The research methodology that has been adopted in this thesis combines the theoretical concepts with experimental evaluation, resulted from simulations through programming. When dealing with scalability and massive size of the real cloud infrastructures, simulations can assist in testing the proposed approaches in a smaller scale environment.

In the first experiment, Java runtime environment 8.0 and Cloud-Sim (Nuaimi K.A., et. al., 2012) have been selected for implementing the MFL-APSO design. Java is a useful high-level programming language which makes simulation much easier by providing the developers with using existing classes and libraries. To implement the proposed replication strategy ADRS-

DDMC Cloud-Sim has been adopted for this research. The code has been modified and provisioning policies have been added.

Reason for the selection of Cloud-Sim simulator:

1. Maintain quality of service
2. Efficient resource utilization
3. Dynamic workload distribution
4. Anticipatory workload distribution
5. Easy for testing and maintained.

Cloud-Sim is an open source simulation tool, which initially has been developed by the University of Melbourne and is useful for implementing the cloud-based case studies. Cloud-Sim is an extensible toolkit that enables cloud users to simulate their provisioning models in a simulated cloud environment.

3.6 RESEARCH RULES

As the aim of this research is to optimize the load balancing methods in cloud-based systems, certain research rules have been followed in this thesis which is described below:

1. Conduct a systematic literature review to study the previously proposed methods. This step helps to understand the general concept of the load balancing combined with investigating the existing limitations within other proposed load balancing approaches in cloud-based systems.
2. Architect a theoretical model to solve the load balancing issues observed from step 1. This step highlights the focus of the thesis by formulating the hypothesis and developing the research questions.
3. Model and validate the theoretical model through programming and analyses the outcome results. This step designs the experiments needed to validate the theoretical model followed by implementations procedure. The expected outcomes could validate the hypothesis.

3.7 MODELLING APPROACHES

In this research work, the serial gray-box approach has used which is initiating with the black-box approach by modeling and pre-processing the internal elements needed for designing the system. The data then outputted toward white-box modeling. The serial gray box modeling

approach is known as general regression model which applies to the systems that their details are not clear and needs further improvements.

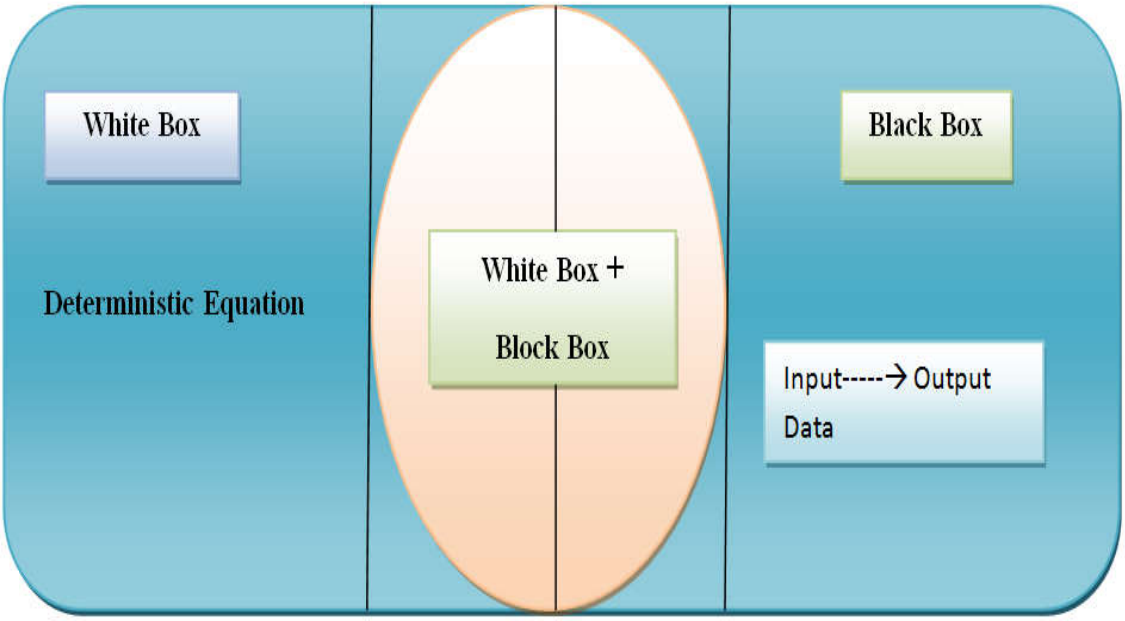

Figure 3.7.1 Grey Box Modeling Approach

An example of this could be explained as obtaining the task load balancing structures and highlighting the overall system health status such as bandwidth, memory rate. These elements could be captured with black-box elements and outputted toward white box modeling approach to estimate the time span needed to schedule the tasks. Then the estimate can be used to monitor the task load balancing and control the resources needed to optimize the system's load balancing. Figure 3.7.1 clearly shows in grey box modeling approach where both black-box and white-box models are combined in the proposed method.

3.8 EXPERIMENTAL METHODOLOGIES OVERVIEW

In this research work following methods are used:

3.8.1 Advanced Anticipatory Approach-In this research, the experiments are designed based on advanced anticipatory behavior concept. The purpose of the advanced anticipatory system is to have a predictive model which is aware of the system and the environmental changes. With this awareness, the system can anticipate the future changes and thus apply proper behavior to

adapt instantly to the coming changes. The details of the experiments are illustrated in the section below.

3.8.2 Modified Fuzzy Logic Based Model for Cloud Computing-In this cloud computing research work, the experiments are designed based on Modified Fuzzy Logic Based Model. A fuzzy logic is an approach to computing based on, **degrees of truth** rather than the usual, **true or false**, or **1 or 0**; the modern computer is also based on this Boolean logic based.

The proposed MFLM model is formulated as knowledge base fuzzy expert system modeling. It is based on a novel approach that has been tightening in the data center to find the new perception called Data Center Load Efficiency (DCLE). This factor is predicted in network load configuration region. The factors are Bandwidth (BW), Memory (MEM) and CPU cycle of the data center. This important knowledge of finding a DCLE is mentioned in terms of fuzzy inference rules which connect antecedents with consequences.

CHAPTER 4

MFL-APSO MODEL FOR CLOUD COMPUTING

In this chapter, a new load balancing for cloud performance improvement is introduced which is based on **Modified fuzzy Logic and Advanced Particle Swarm Optimization Model (MFL-APSO)** to optimize the total execution time of tasks in the workflow applications. The key objective of applying the MFL-APSO method is to minimize the total tasks execution time by verifying the load fluctuations of the interconnected tasks. The variances of the algorithm consider factors such as load variations (fluctuations) and optimization of the data retrieval time.

4.1 INTRODUCTION

This chapter describes the design and working of the heuristic algorithm that uses Modified fuzzy Logic and Advanced Particle Swarm Optimization Model (MFL-APSO) (described in previous chapter 3). The model optimizes the load balancing method by minimizing the total execution time. In the cloud architecture, in order to manage the allocated tasks, there is an application scheduler designed to balance the workload between available resources. By analyzing the memory usage, processing duration and data access time, application schedulers maximize the resource utilization.

As discussed in previous chapters, various algorithms have been designed for optimal load distribution between available computing resources. A PSO method is a load balancing method that was introduced by (Robert, et al., 2013).The algorithm is focusing on the social behavior of the particles in one population. Each particle obtains the best local position and the best global position in the entire population. The proposed MFL-APSO model (figure 4.1) gives an efficient load balancing. The main aim or objective of the proposed method MFL-APSO is to achieve optimum resource utilization and efficient load distribution or balancing between all the computing resources on cloud servers. Initially, the resource utilization of the load is to be derived based on parameters such as memory utilization, processing time, response time and speed. In the proposed algorithm, the current load will be computed for all the resources shared in a virtual machine of cloud servers. Once the resource level percentage is calculated for all the resources in a VM, load balancing operation will be initiated to effectively use the resources

dynamically in the process of assigning resources to the VM to reduce the load value. Utilizing the effective optimization procedure instead of existing HEFT and SHEFT algorithm can lead to better load balancing since HEFT and SHEFT is a weighted priority based traditional algorithm for load balancing. Accordingly, an optimization algorithm called MFL-APSO method to do the load balancing operation is proposed in this work. The resource level percentage will be calculated based on the resource level percentage load assigned to the VMs.

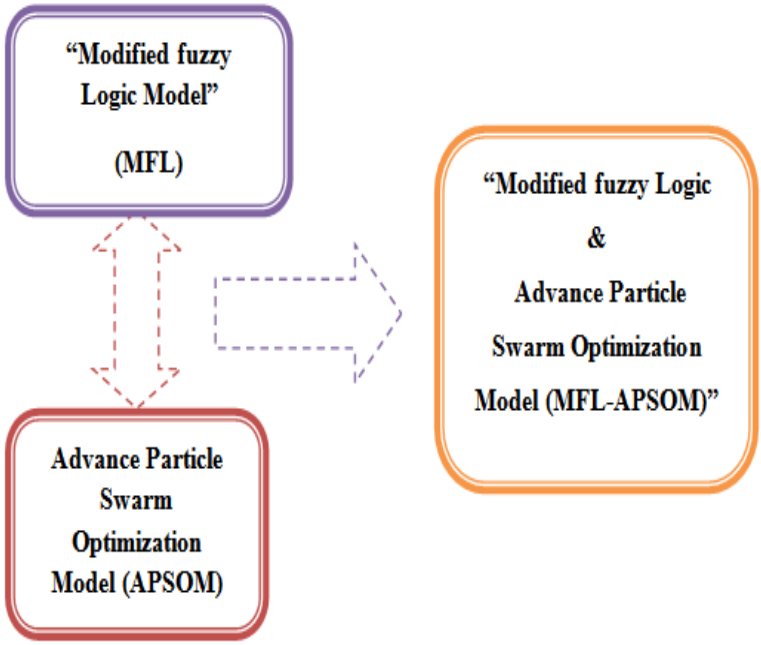

Figure 4.1 Proposed MFL-APSO Model

The primary focus of this chapter is to schedule an efficient load balancing algorithm to improve the performance of the cloud computing system and achieved optimum utilization of computing resources. To enhance an efficient load balancing, it is necessary to overcome the limitations of the existing algorithm. The performance analysis has produced expected results and thus has proved that the proposed approach is efficient in optimizing schedules by balancing the loads. This load aims to project the optimal mapping of the tasks on available resources. The experiment presented in this chapter combines MFL model and APSO algorithm to project the global perception of the workflow application load. The analysis aims to distinguish the fluctuation of the load between dependent tasks in a workflow application. The experimental results confirm the applicability and importance of the proposed MFLM-APSO model in cloud computing domain.

4.2 PROBLEM FORMULATION

In cloud computing, load balancing is one of the major challenges that play an important role in defining the performance of the cloud system. Without having an effective load balancer, some resources will be under-utilized and some of them will be over-utilized. Hence, to design a competent balanced system, main elements such as load estimation, load comparison and interaction between tasks and resources should be considered.

As mentioned earlier (chapter-3), the main objective of our experiment is to minimize the total workflow execution time which includes minimizing the make-span element by considering the load fluctuations between dependent tasks. In order to formulate our optimized load balancing model, workflow application is examined as a directed acyclic figure as shown in figure 4.2.1. In the presented figure 4.2.1, the nodes define the tasks and the edges denote the dependencies between tasks and their neighbors. Task T1 is a root task. It generates input data for task T2 the child task.

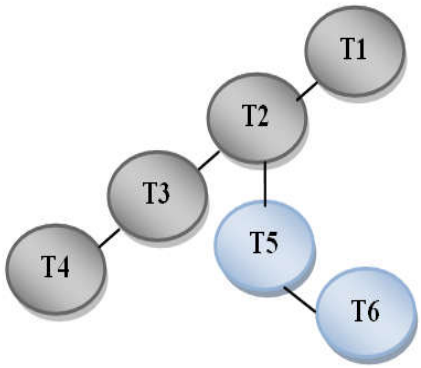

Figure 4.2.1 Example of Workflow Modeling

The model of the algorithm is formulated using the following parameters-

- In figure 4.2.1 shows as G=(V, E), where V={1,2 …n} is the set of tasks where n is the number of tasks. E= {LWij} denotes the load weight and the information exchange between task i and j.
- DLij presents the load weight between task i and j. We consider BWij i, j= {1, 2, 3 …s} as the bandwidth value between two nodes, where s is explaining the number of nodes.
- Pij= {1, 2… k} represents k computing resources. DP_{ij} indicates the amount of data that the task i assigning to processor j. PC_m and PC_c are the processor memory and processor capacity. Based on the introduced value in above section we calculate-

4.2.1 Working of MFL-APSO Model-In cloud computing, VM is not shared by multiple users based on the user requirements. The physical resource is converted to virtual resource and allocate to a single user. Figure 4.2.1.1 shows working in Proposed MFL-APSO Model.

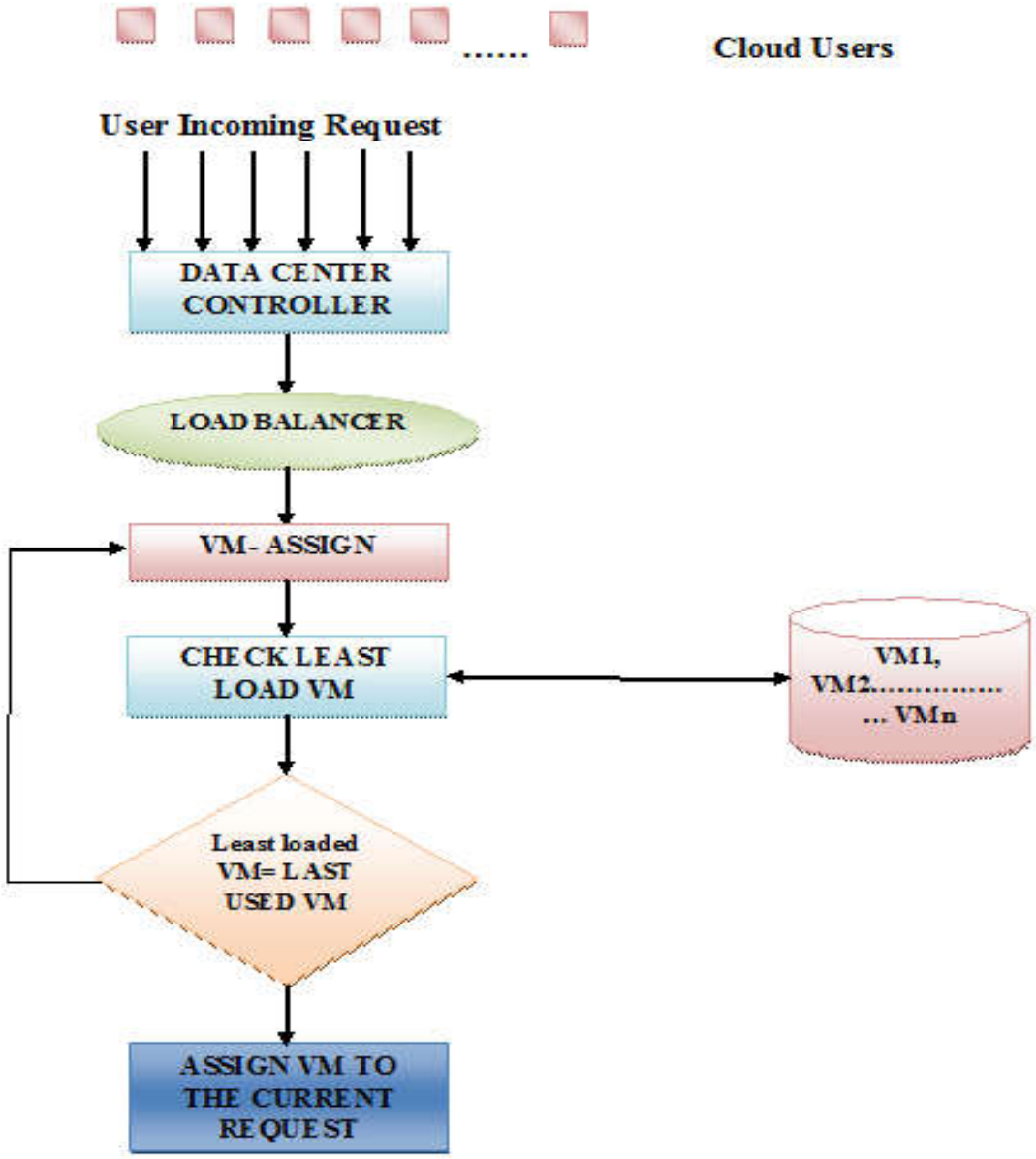

Figure 4.2.1.1 Working of Proposed MFL-APSO Model

The proposed MFL-APSO, algorithms firstly compute the current load for all the computing resources of a virtual machine of cloud servers. After successful measuring the load for all the computing resources, load balancing method efficiently utilizes all the available resources dynamically into the allocation and assigning of computing resources to the particular node to

achieves less or better load value. So, for assigning of resources to properly VMs with fuzzy rules, an MFL-APSO is suggested for load balancing. A load balancing is done within the VM based on the processors. Each processor can share the memory based on the memory allocated to the VM and not to the physical machine. The proposed MFL-APSO method uses three processors VMs. Initially, the user load is sent to the queue, the load balancer gets the load from the queue and gets the status of the processor in a VM to allocate the load to a particular user VM.

The load is split only inside the VM based on the number of processors in the VM. For each processor, the load level is calculated by using the resource level percentage parameter. The proposed methodology is more efficient than the existing algorithm by this parameter. The efficiency is proved by the increase in speed and decrease in response time and execution time. Load balancing is a major research issue in cloud computing. Load balancing is a process to distribute the dynamic workload across multiple virtual machines in a balanced way. The major benefits of load balance are: (i) it optimally utilizes the computing resources and (ii) increases the performance of CC.

4.2.2 Parameters of Proposed MFL-APSO Model-Proposed performance method use performance parameters as a key component for comparisons. Table 4.2.2.1 shows Parameters of Proposed MFL-APSO Model.

Parameters of Proposed MFL-APSO Model

Method	Performance Parameter for
Modified fuzzy Logic and Advanced Particle Swarm Optimization Model (MFL-APSOM)	➢ Execution Time ➢ Response Time/ Latency ➢ Resource Utilization ➢ Data Transfer Rate ➢ CPU Load & CPU Time ➢ Memory Rate ➢ Workflow and Soft error rate

Table 4.2.2.1 Parameters of Proposed MFL-APSO Model

The load balancing algorithms analyze the following metric in the proposed method. In order to achieve more optimized load balancing algorithms, it is necessary to utilize resources efficiently. Many load balancing algorithms in Cloud computing and their challenges are reviewed in the previous chapter. Based on the related work, some of the issues of the existing algorithms are identified: less speed, more execution time and sequential process. The limitations can be suppressed by modifying the procedure of resource level percentage and fuzzy rules with the new proposed algorithm.

4.2.2.1 Latency or Response Time-(Disha, et al., 2013) suggest that the latency rate can be calculated by using the status of the existing load in the VM. All the existing job status, such as allocated time, starting time and finishing time is taken. The total end to end network latency is defined as the sum of complete delays of all the links between the user and the VM. The interface delay depends on the various sizes of packets which transmitted and varies with network bandwidth and link. It is equal to the size of the data unit divided by bandwidth. The latency rate is calculated as-

$$\textbf{Latency\_Rate} = \sum^{n}_{t=0} (CTt - ATt) - ETt / n \quad \text{----------4.2.2.1.1}$$

Where n = Number of jobs

4.2.2.1.1 Need for Latency or Response Time Parameter-For measuring the storage performance in cloud computing, latency is one of the important parameters to analysis the VM job status.

4.2.2.2 Execution Time and Task Transferring Time-(Ravi K. S., et al., 2013) suggest that the execution time can be calculated by measuring the time taken for completing the process time of virtual machines in the cloud data center. The execution time is the total amount of time the virtual machine takes in executing the instructions. The total time of a running program is much lesser than the total execution time of the program.

- **Execution Time,** $ET = \sum^{n}_{t=0} (FT_i - ST_i)$ ----------4.2.2.1.2
- **Mean Execution Time,** $ET_{mean} = \sum^{n}_{t=0} (FT_i - ST_i) / \sum^{n}_{t=0}$ ----------4.2.2.1.3
- **Task execution Time,** $T_{exe} (M) = \sum^{n}_{i=1} \sum^{m}_{t=1} X_{it} * DP_{it} / PC_m * PC_c$ ----------4.2.2.1.4

Where T_{exe} (M) denotes the task execution time where X_{it} is 1 if task i is assigned to processor t otherwise 0.

- **Total task Transferring Time-** The total task transferring time can be presented as equation (4.2) where Y_{ijkl} is 1 if task i is assigned to processor k(where k!=1) and task j is assigned to is processor L for load balancing otherwise 0.

$$\mathbf{T_t\ (M)} = \sum^{n}_{i=1} * \sum^{m}_{k=1} * \sum^{m}_{l=1} * Y_{ijkl} * DLit / BW_{ij} \qquad \text{----------4.2.2.1.5}$$

4.2.2.2.1 Need for Execution Time and Task Transferring Time Parameter-For measuring the cloud traffic, execution time is the best important parameter to load in fast and analysis the VM speed.

4.2.2.3 Resource Level Percentage-Resource level will be combined by using the three parameters like a number of processors, load with latency rate and also use execution rate. The first part represents the number of processors (P) indicates the maximum number of processors in the cloud environment in a single virtual machine. The current load will be increased when the number of jobs allocated improves. The method for calculating resource level percentage is identified by using number of processor P, latency L, execution E, current load cl, than resource level % -

$$\mathbf{RLP} = \sum \frac{\mathbf{P/Max_p + e^{log2\ Cl} + e^{log2\ L} + E}}{\mathbf{N}} \qquad \text{----------4.2.2.1.6}$$

4.2.2.3.1 Need for Resource Level Percentage Parameter-For efficient load balance the proposed methodology to be used is resource level percentage parameter. This parameter analyses the better storage space available for efficient load in VM disk storage.

4.2.2.4 Fuzzy Rules for MFL-APSO Model-The fuzzy controller works as a feedback control system, which repeats the various computing cycles to achieve the desired output. To establish the fuzzy controller modeling, first, we have to identify and define the various inputs and output variables. A performance improvement function is used to calculate the complete system performance, which is calculated by using performance factors-

1. CPU loads such as speed and latency
2. Environment variables

3. Type of computing resource
4. Memory (MEM), Bandwidth (BW) and CPU Cycles (CPU).

The result of the resource level percentage is an efficiency parameter in the proposed method. The resource level percentage is predicted by fuzzy rules as low, medium and high.

The fuzzy rules for resource level percentage rate are as follows:

1. Memory utilization is low and then resource level percentage is high.
2. Memory utilization is medium and then resource level is medium
3. Memory utilization is high and then resource level is low.

The efficiency is identified based on the speed of the VM. The speed is calculated using latency rate. The average latency is taken in every VM. Based on the average latency the VM speed is analyzed. Table 4.2.2.4 reveals the analysis results of fuzzy parameter including network traffic, disk storage and RAM. Using this parameter (section 4.2.2.4), the storage of the cloud data is calculated as higher, slower and medium.

The latency level is assigned through fuzzy rules as high, low and medium.

a) If (CPU load is low) and (memory utilization is low) then (resource level is medium)
b) If (CPU load is low) and (memory utilization is medium) then (resource level is low)
c) If (CPU load is low) and (memory utilization is high) then (resource level is low)
d) If (CPU, load is medium) and (memory utilization is low) then (resource level is high)
e) If (CPU load is medium) and (memory is medium) then (resource level is medium)
f) If(CPU load is medium) and (memory utilization is high) then(resource level is low)
g) If (CPU load is high) and (memory utilization is low) then (resource level is high)
h) If(CPU load is high) and (memory utilization is medium) then (resource level is medium)
i) If (CPU load is high) and (memory utilization is high) then(resource level is medium)
j) If (network traffic is low) and (storage disk is high) and (primary memory RAM, is high) then (performance is high)
k) If (network traffic is average) and (storage disk is average) and (primary memory RAM is average) then (performance is average)

l) If (network traffic is high) and (storage disk is low) and (primary memory RAM is low) then (performance is low).

Fuzzy Parameters For Proposed

Network Traffic	Disk Storage	RAM	Performance in Cloud
Slower	Higher	Higher	Higher
Slower	Slower	Slower	Medium
Slower	Slower	Higher	Medium
Slower	Higher	Slower	Medium
Slower	Medium	Higher	Medium
Slower	Medium	Medium	Medium
Higher	Higher	Slower	Medium
Higher	Medium	Slower	Medium
Higher	Slower	Slower	Slower
Higher	Medium	Medium	Slower

Table 4.2.2.4 Analysis Results of Fuzzy Parameters

The fuzzy parameter analyses the network traffic. Based on the network traffic, the speed such as execution time and the response time is calculated.

4.2.2.4.1 Need for Fuzzy Parameter-One of the major Advantages of cloud performance can be improved due to the availability of resources in the cloud. The performance of network traffic in the cloud can be improved by taking care of parameters such as latency and execution time and resource level percentage. When network traffic is Slower than the performance of the system will be Higher, when the disk storage and RAM usage is Higher than the performance will improve.

Performance Network Traffic

- Higher <300 MB
- Average= 300 MB
- Slower >300 MB

- **Performance Disk Storage**
  - Slower <2 GB
  - Average= 10 GB
  - Higher >10 GB

When the network traffic is higher, even when disk storage and RAM are higher or medium, the final performance is slower or medium. When the network traffic is slower in spite of disk storage and RAM being slower or medium, the performance is higher or medium. Thus network traffic plays a major role in the performance of cloud.

4.2.2.5 Average VM Resource Utilization-The mathematical approaches for average resource utilization of all VM in a data center are calculated to be slower. Where n is the total number of CPU in VM.

$$CPU_{VRU} = \frac{\sum_{i}^{n} CPU_{i}^{u}}{\sum_{i}^{n} CPU_{i}^{n}} \qquad \text{----------4.2.2.5.1}$$

Similarly, the average utilization of memory, bandwidth of VM in a data center is -

$$MEM_{i}^{u}, NET_{i}^{u}, MEM_{u}^{A}, NET_{u}^{A} \qquad \text{----------4.2.2.5.2}$$

The load information is taken from current CPU utilization, memory utilization and network traffic of a VM Each metric is calculated in mathematics equation (1)

$$Load\ n = \Sigma\ (cpu\_un, mem\_un, n\_un) \qquad \text{----------4.2.2.5.3}$$

Where Load n= Load value of n VMs, cpu_un = Average of CPU utilization, mem_un = Average of memory utilization, n_un = Average network traffic.

4.3 ALGORITHM FOR PROPOSED MFL-APSO MODEL

The new proposed algorithm for load balancing is based on Modified Fuzzy Logic and Advanced Particle Swarm Optimization Model, to optimize the total execution time of tasks in the workflow applications. It is based on a heuristic algorithm that uses Modified Fuzzy Logic and Advanced Particle Swarm Optimization Model.

In the existing system, HEFT is widely used. HEFT is a heuristic load balancing algorithm that can be applied for load balancing a set of dependent tasks on available resources. To complete the load balancing of a workflow application, HEFT considers both execution time and communication time between each connected tasks. Generally, HEFT algorithm gives priority to

all the workflow tasks based on their finishing time. When all the tasks are prioritized, the task with the high rest priority will be scheduled on the first available resource. Proposed MFL-APSO load balancing algorithm analyses the virtual machine and is able to do efficient load balancing at the time of critical situations. The MFL-APSO algorithm aims to form an efficient load balancing with the use of four parameters such as latency rate or response time, CPU load, mean execution rate and the resource level in percentage.

Proposed MFL-APSO Algorithm for Efficient Load Balancing

Input: Number_of_Resource, Number_of_jobs, Status, Memory, Latency rate, Execution time,

Output: workflows, Allocated resource [], Rate, Jobs []

1. Call the Procedure Load_Balace ()
2. Assign variable and check the condition Job J<-Jobs[i];
3. Check the job status by Current_Job_Length <- J.Length;
4. Assign job, Current_Job_status = J.status;
5. For any instance, If (Current_Job_status == null) then
6. Assign resource by, ResoureLevel [] RL= new ResouceLevel [ ];
7. Repeat step For (k=0; k<Number_of_resorce; k++) then
8. Get resource for the resource by, Resource r = GetResorce ();
9. Assign value to, Processor p= r.GetProcessor ();
10. Calculate job latency by, Latency l = r.GetLatency ();
11. Execution e = r.GetMeanExectionRate ();
12. Calculates system load, Current Load cl = r.GetCurrentLoad ();
13. Set fuzzy by, RL[k] = CalculateRateFuzzy (p, l, e, cl);
14. Kill the process, End For k;
15. Arrange the items by, SortList (RL);
16. Allocates the resources as per order, Allocated_Resource[i] = SortList(0);
17. Kill the process, End For i;
18. Check pending process
19. If no pending process found than End
20. Else complete the pending process
21. End

4.3.1 Steps for (MFL-APSO) Model-Following steps are used

Step-1 The load balancer receives the data from cloud service providers. The load balancer is a middleware intermediate to CSP and cloud data center.

Step-2 The load balancer processes the data center and identifies the number of VMs with a number of processors in each VM. The load balancer gets properties of VMs such as total memory, used free, memory available memory, latency rate, response Time and execution time is calculated.

Step-3 Resource level percentage is calculated for each VM using this property. Data center controller manages user request as per cloud recourses.

Step-4 By applying the fuzzy rules, the resource level status is identified as slower, medium and higher.

Step-5 The load is allocated based on the percentage identified by the resource level percentage metrics. This RLP (Resource Level Percentage**)** is an efficiency parameter in this proposed methodology to increase the efficiency of the cloud computing.

4.4 SIMULATION AND RESULTS ANALYSIS

4.4.1 About Simulation-To evaluate the performance of our proposed algorithm Java programming has been used to extend the Cloud-Sim simulation tool (described in chapter 3).

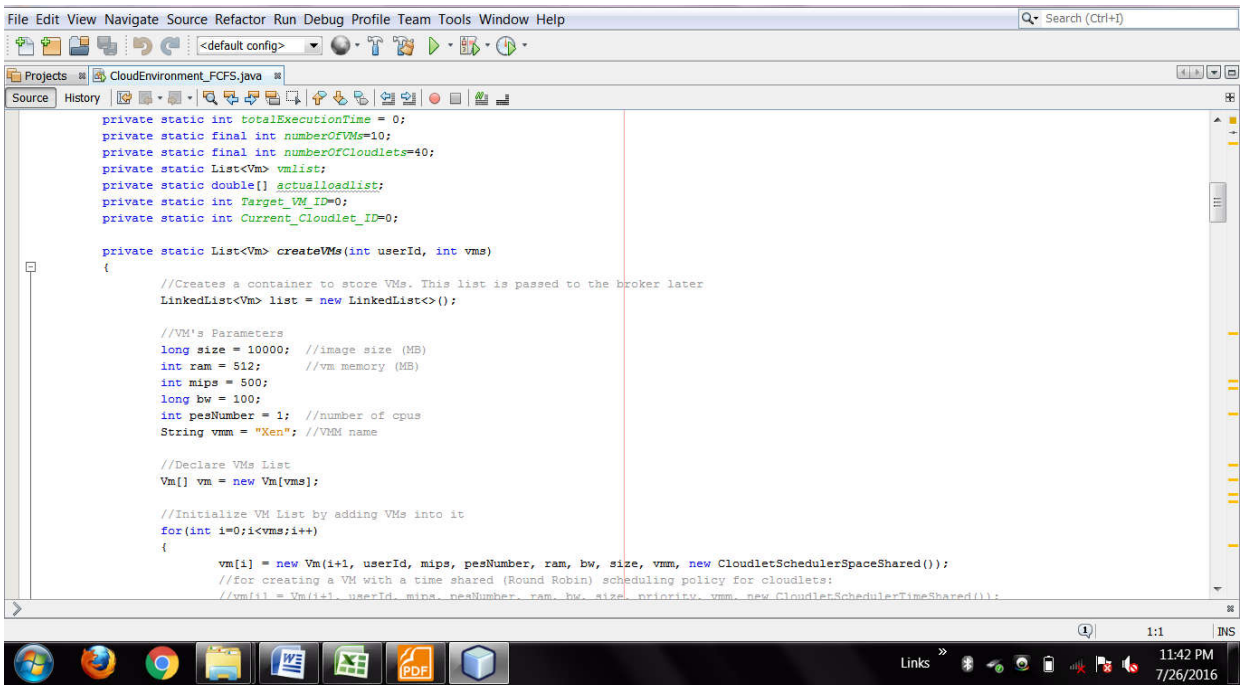

Figure 4.5.1 Creating Virtual Machines for Proposed Model

The broker is responsible for load balancing the requests on available resources. Each site has a replica manager that handles the automatic replica creation and deletion. Different jobs could be submitted to the broker to be assigned to available resources.

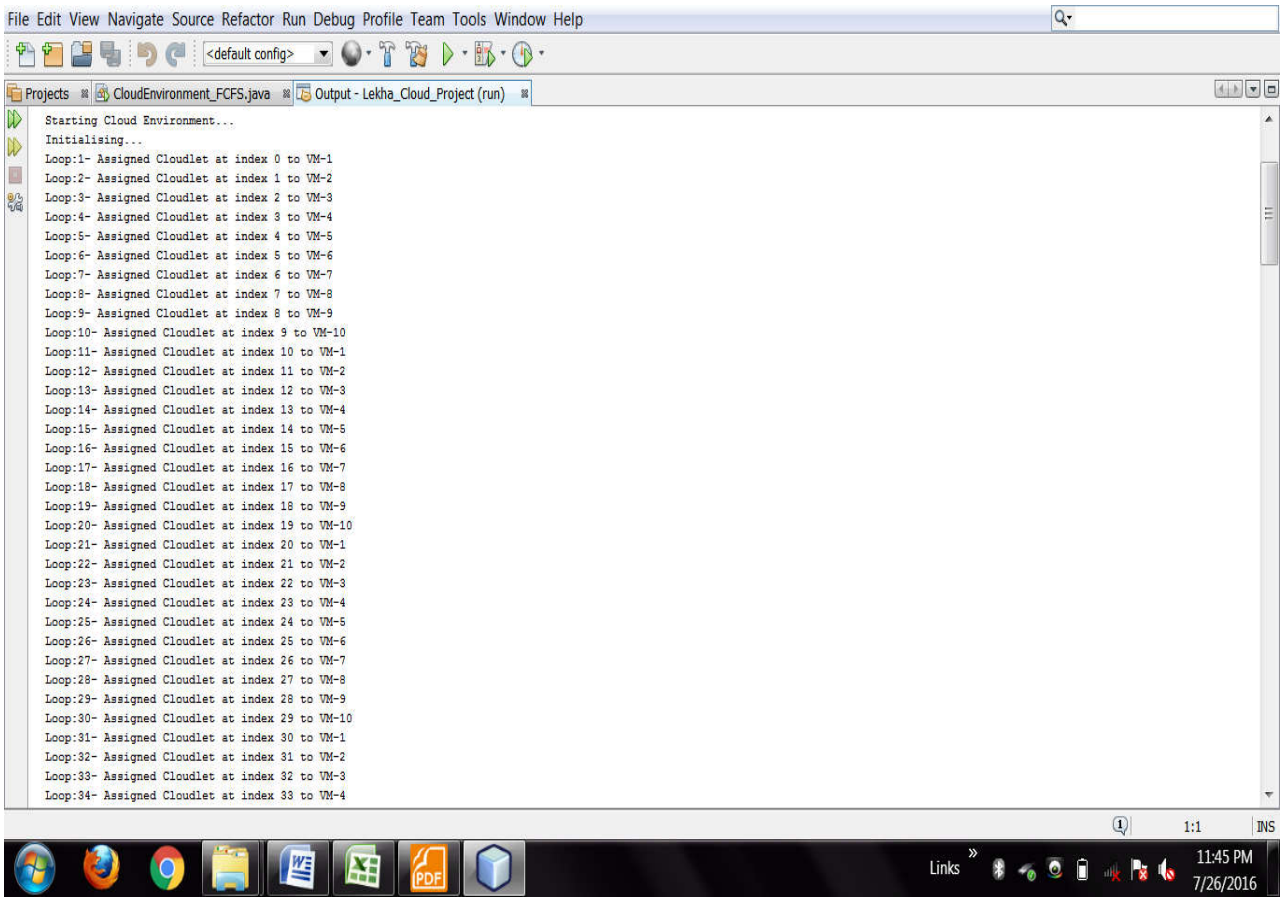

Figure 4.5.2 Creating Virtual Machines at Run Time in Cloud-Sim Simulator

The proposed algorithm uses 300 VMs in a single data center. The memory for each VM can share the physical resources on a server. The different type of file size is given and the files allocated based on the resource level percentage taken in allocated VM. The proposed method is more efficient than the existing algorithm. By increasing the number of VMs, the proposed method is more efficient.

4.4.2 Simulation Environment-A fuzzy rule is designed in the proposed algorithm. These rules determine the performance of the network and the VMs as slower, medium and higher. The rules are used to identify the performance of the VM in the cloud. The efficiency is proved based on the performance of the VM such as average response time and average execution time of the VM

in a data center. The algorithms (proposed and existing) have been implemented in Cloud-Sim and the simulation results give better performance than the existing algorithm.

The response time of the VM is reduced and the latency rate is also reduced by using the proposed methodology. The simulation analysis has taken 300 virtual machines with three processors with different computing cycle. For every computing cycle, a load of each processor in the VM is increased and decreased to get the different results. For each computing cycle, the speed, latency, memory utilization and the response time give better performance. For every computing cycle, the VM is increased gradually and results in better performance. This experiment gives efficient load balancing with the proposed algorithm MFL-APSO**.** The simulation result is obtained from the parameters latency, response time, execution time and resource level percentage. During the performance evaluation, the parameters used in the proposed MFL-APSO, lesser CPU utilization rate and the memory usage rate in percentage. By considering the performance schedule is generated by balancing the given load.

In performance analysis, the maximum utilization of CPU rate of each VM can perform differently in the proposed MFL-APSO algorithm. The analysis and complete study show that how efficiently and effectively the proposed method performs under various load levels of CPU rate. The maximum utilization of the CPU varies for each VM. The load balancing in different CPU is represented by the response time and execution time of the current load in the VM. The memory utilization is calculated as the utilization of the memory in the VM. The percentage of memory is taken in each VM and the load is allocated based on the percentage level. The CPU, memory utilization and processing time are considered as the loads to get the resource level percentage. The load balancing parameter should be designed in such a way that the VM will be selected for a load with the percentage allocated to the VM.

The comparative analysis of the proposed MFL-APSO approach with the existing HEFT and SHEFT approach gives the expected result. The proposed approach utilizes the CPU rate more efficient than the existing approach under the load balancing condition. So, in the load balanced condition and the CPU utilization rate, the proposed approach has better efficiency over the existing approach.

4.4.3 PARAMETERS CALCULATED FOR EXISTING AND PROPOSED METHODS

Following parameters are calculated for existing HEFT, SHEFT method and proposed MFL-APSO method.

4.4.3.1 Response Time Analysis-The proposed MFL-APSO helps to determine the overall allocation of resources with the help of the resource level percentage parameter. The proposed methodology completes optimality check which results in reducing the processing and response time.The execution time is to execute any task over the cloud is also reduced. For any load balancing method lesser response time shows better performance. A numerical analysis of for response time results of the proposed algorithm and existing algorithm clearly shows that the proposed algorithm reduces the effective average execution time of all requests and reduces the average response time.

Response Time for Proposed MFL-APSO & Existing Methods

Virtual Machine VM	Average Response Time (Milliseconds)		
	Existing HEFT Method	Existing SHEFT Method	Proposed MFL-APSO Method
10	37713.15	36047.22	35077.09
20	28956.83	25474.14	21365.22
30	18665.23	17878.44	16342.32
50	14345.56	13245.57	12560.26
75	12456.34	11998.43	10986.89
100	10976.23	9978.25	9809.78
125	8964.67	8547.77	7905.65
150	7980.45	7475.35	6890.46
200	5897.98	5457.21	5256.67
250	4789.78	3744.89	2985.91
300	3745.88	3447.08	2144.52

Table 4.4.3.1 Response time for Proposed MFL-APSO & Existing HEFT, SHEFT Methods

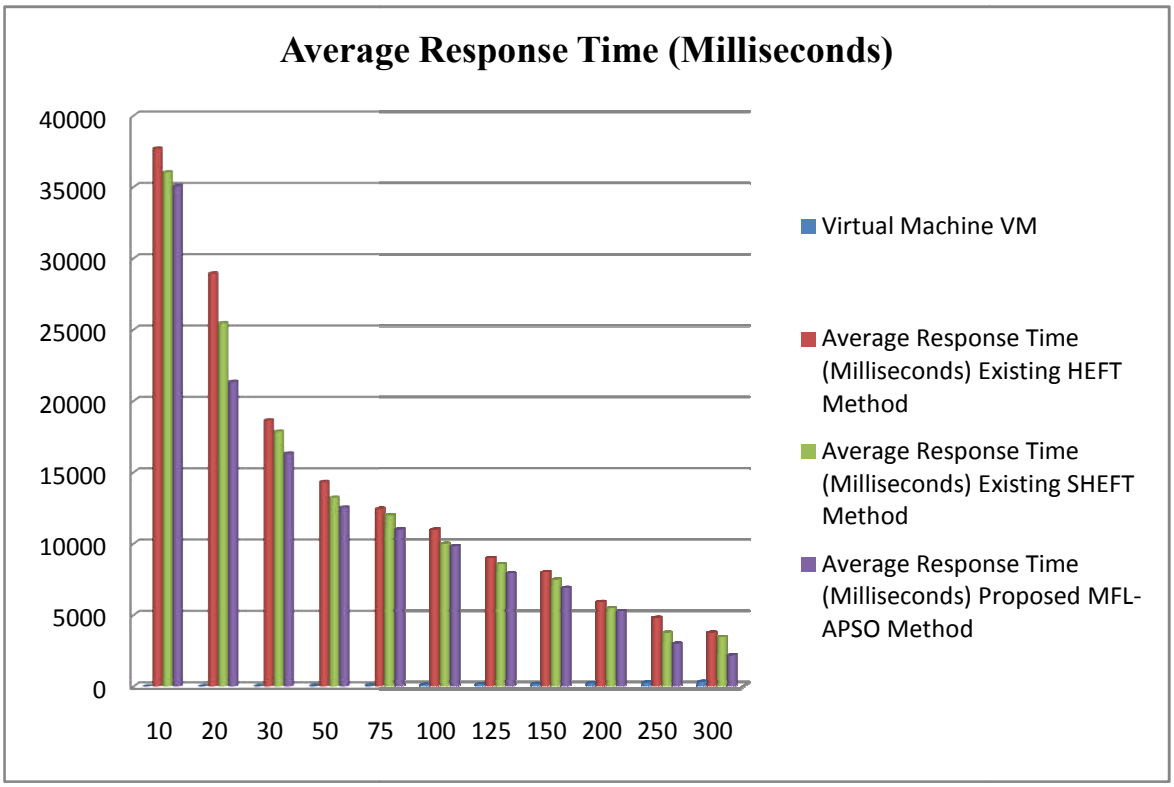

Figure 4.4.3.1 Response Time for Proposed MFL-APSO & HEFT, SHEFT Method

Its shows comparisons of the average response time for the 300 VMs and the result shows there is a decrease in response time. Lesser the response time shows better performance and proposed method shows efficient results than the existing methods.

4.4.3.2 Execution Time Analysis-Table and figure 4.4.3.2 shows the comparative analysis of average execution time of the 300 VMs.

Execution Time Analysis for Existing & Proposed Method

Virtual Machine VM	Average Execution Time (Milliseconds)		
	Existing HEFT Method	Existing SHEFT Method	Proposed MFL-APSO Method
10	30789.98	29877.45	28745.22
20	22478.45	21447.25	19657.25

30	18477.44	17887.47	16447.33
50	14878.57	13554.25	12441.14
75	13787.11	11224.27	10987.55
100	10748.25	9874.55	8945.78
125	8247.17	7748.99	7014.36
150	6874.96	6547.14	5879.54
200	4325.12	3377.48	2987.66
250	3247.85	2544.66	2103.66
300	1787.48	1854.33	1455.65

Table 4.4.3.2 Execution Time for MFL-APSO& Existing HEFT Method

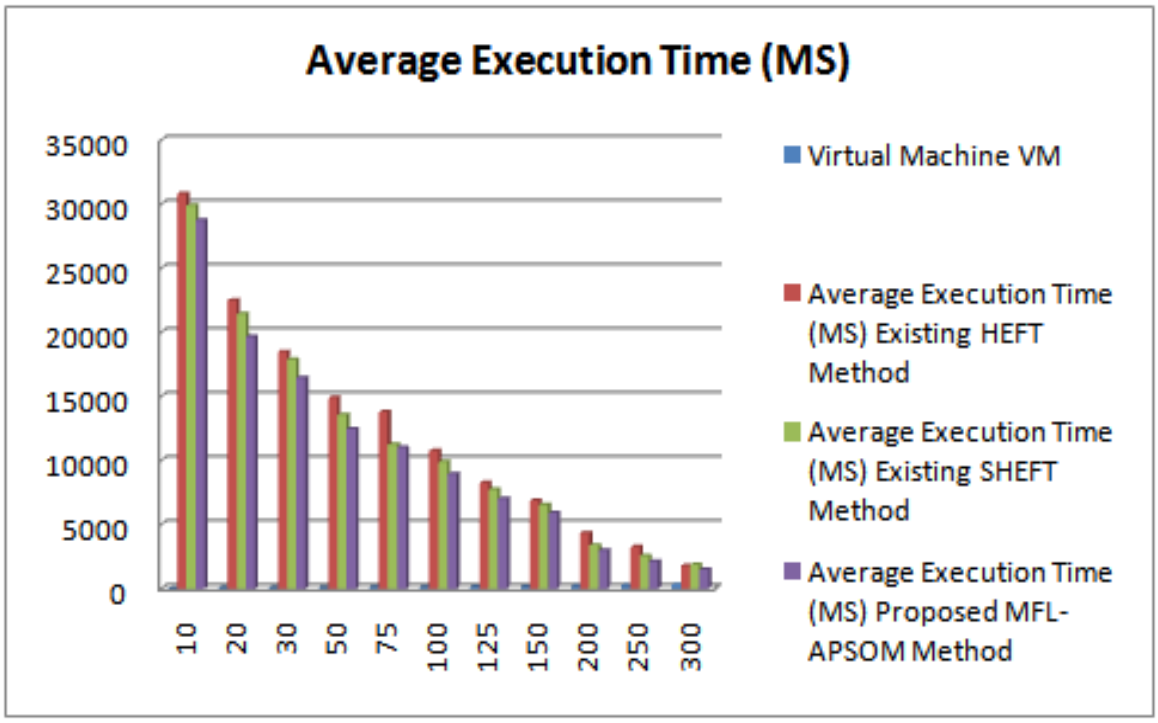

Figure 4.4.3.2 Execution Time for MFL-APSO SHEFT & HEFT Method

In the figure, 4.4.3.2 graph is plotted between the number of VMs (X-axes) and average execution time (Y- axes). Its shows comparisons of the average execution time for the 300 VMs and the result shows there is a decrease in execution time. Lesser the execution time shows better performance and proposed method shows efficient results than the existing methods.

4.4.3.3 Analysis for Data Transfer Rate-Higher data transfer rate shows better performance for a load balancing method. While compared proposed MFL-APSO algorithm with the existing HEFT and SHEFT algorithm following results are calculated (table 4.4.3.3).

Data Transfer Rate for MFL-APSO, HEFT & SHEFT Method

Virtual Machine VM	Data Transfer Rate/ Speed (MIPS)		
	Existing HEFT Method	Existing SHEFT Method	Proposed MFL-APSO Method
10	23744	25878	31478
20	31547	33478	39887
30	38745	41245	48795
50	45789	48779	55687
75	52478	55478	66478
100	60457	64558	75478
125	68997	72485	83456
150	75489	78996	83654
200	83569	85969	93668
250	94587	96998	104785
300	97899	104570	110078

Table 4.4.3.3 Data Transfer Rate for MFL-APSO, HEFT & SHEFT Method

In the table, 4.4.3.3 results of data transfer rate are calculated for virtual machines from 10 to 300 for proposed MFL-APSO and existing HEFT and SHEFT method. In cloud computing, better data transfer rate achieved by any method shows better performance of the system.

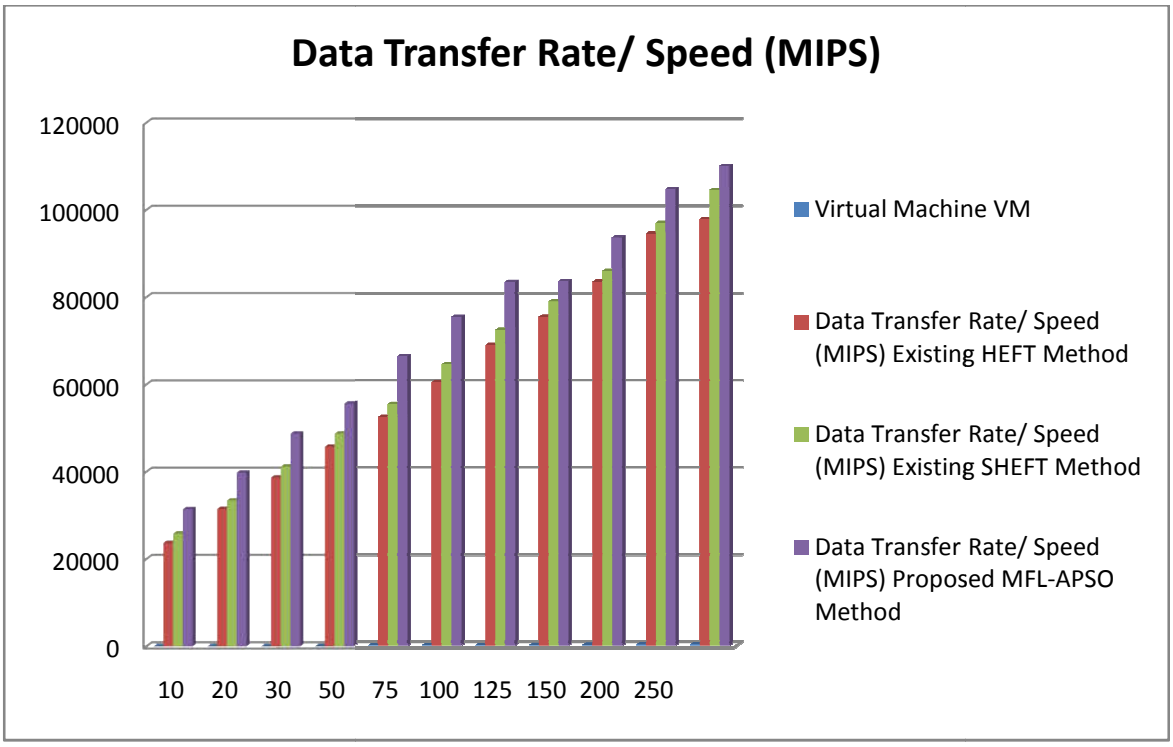

Figure 4.4.3.3 Data Transfer Rate for MFL-APSO, HEFT & SHEFT Method

In the figure, 4.4.3.3 graph is plotted between the number of VMs (X-axes) and Data transfer rate in million instructions per seconds (Y-axes). Figure 4.4.3.3 reveals the comparison of the speed of data transfer for the existing and proposed algorithm. The speed is taken from 300 VMs in different computing cycles. The proposed algorithm gives increased speed at the time of data loading in the VM after the percentage of the resource is identified. The numerical analysis of the speed gives the increased result of the proposed algorithm.

4.4.3.4 CPU Load and CPU Time-This is the time which consumes a process during load and execution of a task. We have calculated the distribution of the workflow tasks onto three different available resources VM 1 to VM 3 for 10 different data size items (64 to 6144 MB) and various fuzzy rules (slower, medium and higher). Table 4.4.3.4 is showing the details of the CPU utilization of the workflow tasks applying the Proposed MFL-APSO algorithm. For any process, less CPU utilization is always desirable for better performance. Proposed method MFL-APSO is designed to achieve lesser CPU utilization. Following results are calculated for CPU utilization % of the existing and proposed method.

CPU Load Utilization Rate Using Proposed & Existing Method

WORKFLOW DATA SIZE IN MB	CPU Utilization In % For Existing HEFT				CPU Utilization In % For Existing SHEFT				CPU Utilization In % For Proposed MFL-APSO			
Fuzzy Rule	Slower	Medium	Higher	Avg %	Slower	Medium	Higher		Slower	Medium	Higher	Avg %
VM	VM1	VM2	VM3		VM1	VM2	VM3	Avg %	VM1	VM2	VM3	
64	19.22	31.88	50.45	33.85	15.44	27.88	47.55	30.29	11.25	24.33	36.22	23.93
128	17.88	34.88	55.47	36.08	14.25	29.85	48.9	31.00	10.25	21.33	38.9	23.49
256	18.66	26.47	53.26	32.80	13.56	24.5	49.77	29.28	9.9	21.36	37.88	23.05
512	16.25	34.98	57.9	36.38	15.88	30.25	44.89	30.34	12.35	26.45	45.66	28.15
1024	15.44	25.44	51.24	30.71	14.87	20.5	47.8	27.72	11.2	20.14	30.25	20.53
2048	18.66	33.25	55.69	35.87	11.25	24.6	44.12	26.66	9.89	20.41	25.6	18.63
3072	17.55	30.15	57.26	34.99	19.78	25.77	44.85	30.13	10.26	25.77	35.25	23.76
4096	16.89	31.22	59.66	35.92	20.11	25.66	34.25	26.67	12.3	26.44	30.21	22.98
5120	19.8	32.14	54.8	35.58	17.25	29.88	42.88	30.00	15.45	25.4	36.55	25.80
6144	17.98	33.45	58.9	36.78	16.45	30.45	40.25	29.05	14.66	24.15	32.32	23.71

Table 4.4.3.4 CPU Load Utilization Rate Using Proposed & Existing Method

Table 4.4.3.4 reveals the load of three VMs using different user file sizes from 64 MB to 6144 MB. The fuzzy parameter gives the performance of the VMs based on the data load. The result of fuzzy method gives the percentage available for loading the data and in VMs is taken at slower, medium and Higher. The simulation result clearly shows CPU utilization % of existing

HEFT, SHEFT and proposed MFL-APSO method for fuzzy rules. Less CPU utilization shows better performance. Proposed method performs outstanding as compared to existing methods.

4.4.3.5 Memory Rate-In this experiment, the maximum memory rate with different values has been set to evaluate the performance of our proposed load balancing model.

In this experiment different categories of memories have been considered:

A. **Cache Memory-** This memory is functioning at the speed of CPU
B. **Physical Memory-** This memory is operating slower than CPU
C. **Virtual Memory-** This memory is the one that we need to check it's rated which are running on the virtual machine and it is much slower than CPU.

Memory Rate for Proposed and Existing Methods

Work Flow Data Size in MB	Memory Rate (Milliseconds)		
	Existing HEFT Method	Existing SHEFT Method	Proposed MFL-APSO Method
64	565	578	661
128	325	345	365
256	778	789	798
512	445	478	487
1024	678	689	702
2048	548	569	588
3072	798	821	834
4096	678	697	728
5120	887	893	902
6144	745	759	789

Table 4.4.3.5 Comparison of Total Memory Utilization Rate between Proposed & Existing Methods

In figure 4.4.3.5 graph is plotted between workflow data size in MB (X-axes) and memory rate in milliseconds (Y-axes). For workflow data size in 60 to 6144 MB, the analysis determines that proposed MFL-APSO algorithm perfumes more efficiently under different load levels.

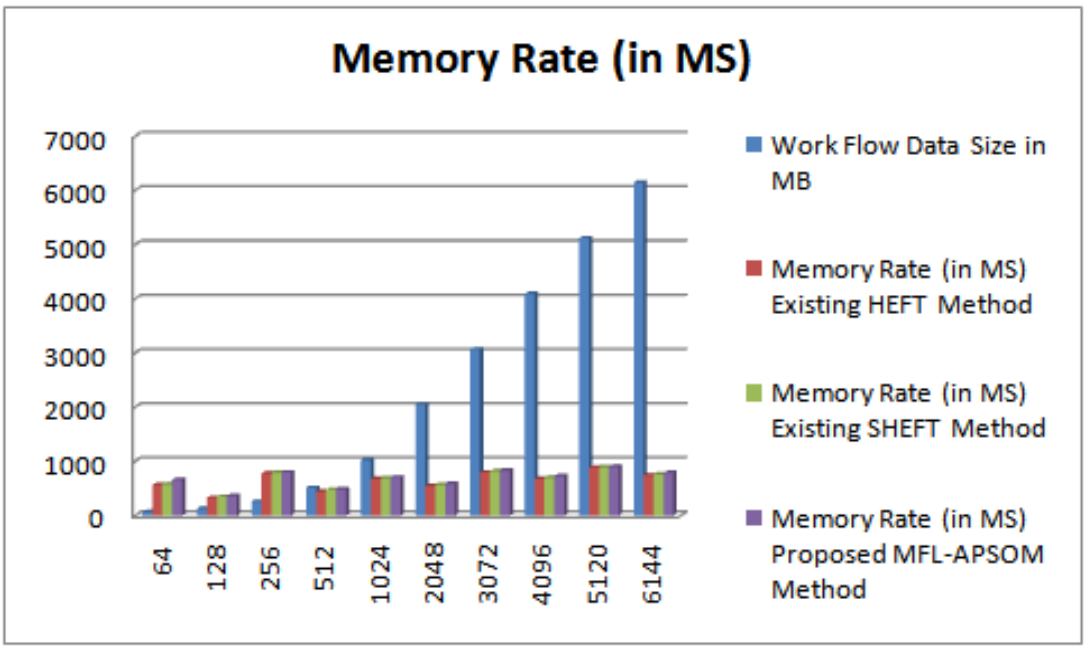

Figure 4.4.3.5 Comparison of Total Memory Utilization Rate of Proposed & Existing Methods

Considering figure 4.4.3.5 clearly shows that the proposed MFL-APSO method utilizes the memory rate more efficiently as compared to existing HEFT and SHEFT methods under load balancing condition for the cloud.

4.4.3.6 Workflow Soft Error Rates-To analyze the fault tolerance of the proposed algorithm we have applied soft errors on system's memory to evaluate the output results of the proposed MFL-APSO experiment. Soft errors usually refer to the events that could change the instructions of the program and data values. In other words, soft errors will have an impact on data only and they won't corrupt the system's hardware. Some of the soft errors could mitigate by a cold reboot of the system.

In our experiment, we have applied system-level soft error to test its effect on output results. These errors usually refer to an event when noises applied on a data bus and corrupt the data that is under process. In fact, the computer read the noise as data bits which can create errors in computing the data inputs. Therefore to apply the soft errors on memory rate we have a tool introduced by (Lie, et al., 2013) to generate soft error rates for selections of the memory blocks. 5% and 10% error rate has been applied to this scenario. 5% error, replicates the 5 events

memory failure for 100 MB memory and 10% error interprets the 10 events memory failure for 100 MB memory.

Impact of Soft Error Rates on Memory

Work Flow Data Size in MB	Soft error, Memory Rate For Proposed MFL-APSO (in %)	
	5%	10%
64	0.61	0.52
128	0.42	0.39
256	0.48	0.44
512	0.42	0.38
1024	0.64	0.57
2048	0.72	0.69

Table 4.4.3.6 Impact of Soft Error Rates on Memory

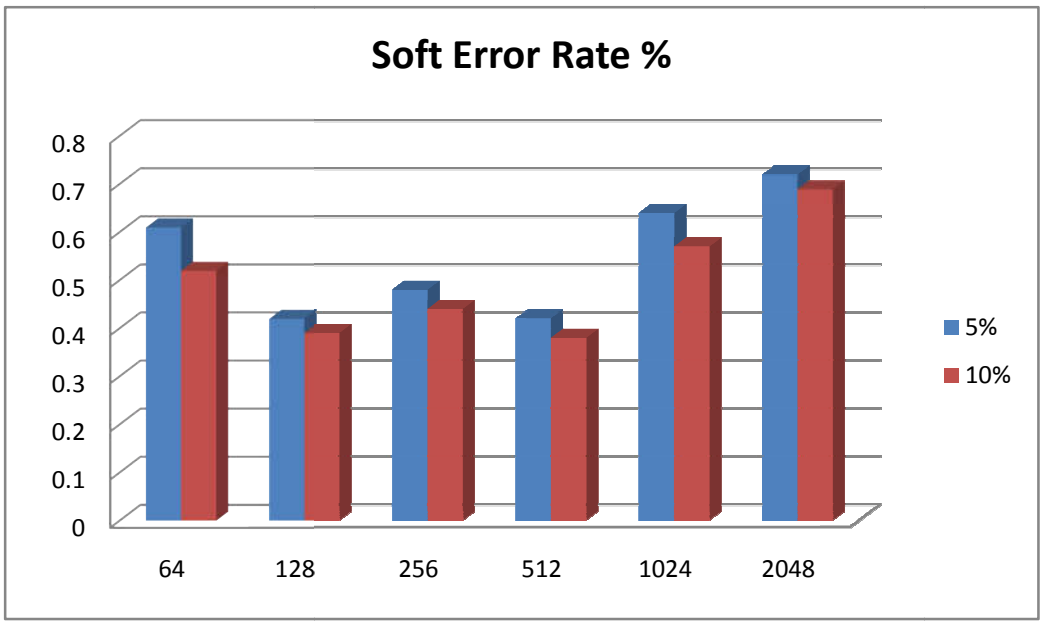

Figure 4.4.3.6 Analyzing the Impact of Soft Error Rates on Memory Rate

In figure 4.4.3.6 graph is plotted between workflow data size in MB (X-axes) and soft error rate % (Y-axes). Above table 4.4.3.6 shows various soft error rates for different data file size of 32

MB,64 MB, 128 MB, 256 MB, 256 MB, 512 MB, 1024 MB and 2048 MB for memory rate 5 % and 10 %. In table 4.4.3.6 as the results indicate there is a dramatic difference while noise has been applied on memory blocks, therefore as future work of this research, to mitigate these soft errors, new error controllers will be applied to a proposed MFL-APSO method.

4.5 RESULT ANALYSIS FOR PROPOSED MFL-APSO AND EXISTING METHODS

Performance evaluation has been done for the response time and execution time. During the simulation, it is found that a response time and execution time are reduced. From above results, it's clearly found that the speed is increased in the proposed algorithm. During the performance evaluation the variable Parameters of the proposed algorithm, consider both the CPU utilization rate and the memory usage rate. Thus the decision parameters mentioned are considered as the load for the nodes. To consider the performance of the data center to schedule the load by balancing loads.

Following parameters are evaluated for existing and proposed method:

1. **CPU Utilization %-**Less CPU utilization shows better performance. Table 4.4.3.4 shows CPU utilization % of various methods such as HEFT, SHEFT and proposed MFL-APSO for different three VMs. In this experimental analysis total three VMs are used for different categories slower, medium and higher. The experimental analysis clearly shows that the load balancing time is uniform for a different rate of CPU and the memory is also balanced as the rate of CPU increases for the proposed method. By analysis, we can say proposed method performs outstandingly in terms of CPU utilization % over existing HEFT and SHEFT methods.

2. **Memory Utilization %-**Higher memory transfer rate % consumes less memory. Table 4.4.3.5 shows memory rate results for existing HEFT, SHEFT and proposed MFL-APSO methods for different works flow data size varying from 64 MB to 6144 MB. The resulting analysis (table 4.4.3.5) clearly shows proposed MFL-APSO algorithm shows better memory rate % with different load levels as compared to existing HEFT and SHEFT methods. The memory rate varying based on the load allocated to the data center.

The proposed algorithm must possess the following characteristics:

a) Maximum CPU utilization.

b) Maximum throughput.

c) Minimum turnaround time and minimum waiting time.

d) Minimum response time.

e) Maximizing speed

f) Minimizing delay time

The proposed MFL-APSO method gives better results in an efficient manner in a cloud network by concentrating on balancing the loads. The proposed approach deals with a simulated cloud network with a set of requests and servers. A load balancing method is developed in the proposed method. The proposed approach is inspired by the MFL-APSO algorithm, because of the attractive features of the MFL and APSOM algorithm. The proposed approach is developed in three steps, initially a population is generated from the cloud network, the resource level percentage analyzed and then load balancing is done using the MFL-APSO algorithm. The performance analysis produces an efficient result and proves the efficiency in optimizing schedules by balancing the loads. The existing HEFT and SHEFT algorithm are compared with MFL-APSO. The resulting analysis clearly shows that proposed method gives a decrease in response time, decrease in execution time and increase in speed.

The following reveals the comparison of the VM speed, response time and execution time for 200 VMs.

3. Response Time (in MS)-The proposed algorithms for the research work done decrease in Response time

- On 5 VM MFL-APSO (29.212%)
- On 10 VM MFL-APSO (17.072%)
- On 15 VM MFL-APSO (26.321%)
- On 20 VM MFL-APSO(18.907 %)
- On 25 VM MFL-APSO(12.105 %)
- On 50 VM MFL-APSO(12.715 %)
- On 125 VM MFL-APSO(11.813 %)

- On 200 VM MFL-APSO (17.661%)
- On 250 VM MFL-APSO (19.719%)
- On 300 VM MFL-APSO (12.809%)

4. Execution Time-The proposed algorithms for the research work done decrease in execution time-

- On 5 VM MFL-APSO (3.087%)
- On 10 VM MFL-APSO (3.994%)
- On 15 VM MFL-APSO (5.083%)
- On 25 VM MFL-APSO (13.344%)
- On 50 VM MFL-APSO (14.178%)
- On 75 VM MFL-APSO (14.213%)
- On 100 VM MFL-APSO (11.562%)
- On 125 VM MFL-APSO (12.168%)
- On 150 VM MFL-APSO (13.399%)
- On 175 VM MFL-APSO (4.991%)
- On 200 VM MFL-APSO (11.321%)

5. Speed-The proposed algorithms for the research work done increase speed

- On 5 VM MFL-APSO (64.289%)
- On 10 VM MFL-APSO (68.628%)
- On 15 VM MFL-APSO (63.931%)
- On 20 VM MFL-APSO (66.984%)
- On 25 VM MFL-APSO (67.657%)
- On 50 VM MFL-APSO (68.079%)
- On 75 VM MFL-APSO (73.720%)
- On 100 VM MFL-APSO (72.976%)
- On 125 VM MFL-APSO (76.038%)
- On 175 VM MFL-APSO (81.094%)
- On 200 VM MFL-APSO (82.183%)

4.6 COMPARATIVE ANALYSIS OF PROPOSED MFL-APSO ALGORITHM AND EXISTING HEFT ALGORITHM, SHEFT ALGORITHM

For performance analysis of proposed MFL-APSO and existing HEFT, SHEFT methods various comparison parameters (table 4.6) have been calculated.

Comparative Analysis of Existing and Proposed Method

Analysis Values	Existing Methods		Proposed Method
	HEFT	SHEFT	MFL-APSO
Latency	Higher	Higher	Slower
Execution time	Higher	Higher	Slower
Delay	Higher	Higher	Slower
Speed	Slower	Medium	Higher
Efficiency	Less	Medium	Higher
Processing time	1.1	1.12	1.5
Priority-based	Yes	Yes	Yes
Scalability	Medium	Higher	Higher
Performance %	72%	78%	87%
Threshold time limit	No	No	No
Technique used	Heuristic Load Balancing	Advanced Heuristic Load Balancing	Work Flow-Based Optimization
Waiting time	More	More	Less
Nature	Static	Dynamic	Dynamic
Approach	Receiver-Initiated	Receiver-Initiated	Symmetric (sender-receiver)

Table 4.6 Comparative Analysis of Proposed and Existing Method

The above Table clearly shows that that the proposed MFL-APSO algorithm gives better results as compared to existing HEFT and SHEFT method.

CHAPTER 5

Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing

This chapter focuses on Phase-II of the proposed AAP-IMC performance model. Phase-II is based on **Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing (ADRS-DDMC)**. This phase introduces a novel dynamic data replication method that is functioning based on Anticipations to create the pre-replicas for future needs of the sites.

5.1 INTRODUCTION

Cloud computing can process intensive applications and various scientific functions across heterogeneous environments. However, computing scientific applications need the huge amount of data which could pose more loads on the network (Apostol, et al., 2011). Although different resources could be dedicated to complete the tasks; there should be a load balancing mechanism which can distribute the load proportionally between the available nodes. In the performance of cloud computing, correct load balancing is always desirable and challenging, these methods minimize the total access time and also increase the availability. Through load balancing, nodes will be controlled and prevented from overloading and as a result system throughput will be enhanced (Chun-Wei Tsai, et al., 2013).

To solve the load balancing issue, replication approach is suggested as one of the load utilization methods that can minimize the data access time (Fangzhe Chang Ren, et al., 2013). Replication creates several data copies of the existing sites which dramatically impact the load balancing performance. Distributing different replicas among available sites, replication can effectively enhance data availability, system reliability and fault tolerance. Replication is controlling the bottleneck in query processing. When a site is accessing a file remotely for several times, it would be more beneficial and more cost saving to replicate that file for site's local access (Hsinyu Shin, et al., 2012). Additionally, replication can minimize the access time. As the files would be accessed locally, communication cost would be reduced. Therefore integrating load

balancing and replication together could be the essential factors of any cloud systems architecture (Janpet, et al., 2013). Considering replication techniques, it is important to recognize which files should be replicated, when replication should be created and where replicas should be stored (Kundu, A., et al., 2014). Generally, replication could be categorized into two groups of static and dynamic. Dealing with static replication, the replica locations are pre-defined and are unchangeable. Also, the created replicas can't be deleted unless the user deletes them manually.

The static data replication methodologies are not adaptable to any real-time changes; therefore they are not suitable for processing the data-intensive applications. On the other hand, dynamic data replications are more flexible to real-time changes and replica statuses are monitored automatically. Furthermore, dynamic data replications are more beneficial in terms of data access cost. By dynamically replicating the required files across the data centers, data access cost is minimized while data availability is revamped (Nader Zaman, et al., 2014). This chapter is proposing a new replication algorithm **Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing (ADRS-DDMC).** This algorithm monitors the access history catalogue to manage the existing replicas and anticipate the needs for creating pre-replicas for future needs. Analyzing the algorithm's performance the response time, access latency and numbers of replica were dramatically decreased.

5.2 PROBLEM FORMULATION

Replication techniques have a dramatic impact on the performance of the cloud computing systems. However, it could be costly if proper replication methodology is not selected. Therefore it is challenging to understand when replication is necessary, which files should be selected for replication, where the replicas should be stored and how the replicas should be synched with the original files (Davia, et al., 2014).

Least Recently Used method (LRU) and Least Frequently Used (LFU) method are popular heuristic examples of data replication methods that have been applied in various cloud computing systems. Although the LRU method and LFU method can minimize the total data access time these data replication methods are not so much aware of the user's future

requirements (Mohan N., et al., 2014). In order to address the limitations of, LRU method and LFU method, there is a need for designing an advanced efficient algorithm that can anticipate the future needs of the sites and also pre-replicate the data. Pre-replicating the required data, the algorithm should effectively minimize the job execution time and enhance the effective network usage so as a result load balancing would be improved. Given the circumstances, in this research, a novel algorithm has been designed that can predict the future needs of the existing sites. Based on data access catalog, the algorithm is able to anticipate the data with high access probability that could be needed in future.

5.3 THE PROPOSED ADRS-DDMC METHOD

This chapter covers working of the proposed ADRS-DDMC method. The proposed method is based on dynamic data replication strategy to increase the performance of cloud system by increasing the overall reliability and availability of computing resource and cloud data. Pre-replication can increase the data availability and robustness of the cloud systems and hence requested jobs can be completed with minimum execution time and high network usage output.

To find the reliable virtual machines the proposed method uses reliability decision mechanism process. After finding out the reliable virtual machines, the client request may be transferred to that VM for further process. The proposed ADRS-DDMC method is based on dynamic distributed Model and data replication strategy. Figure 5.3.1 shows methods of ADRS-DDMC model.

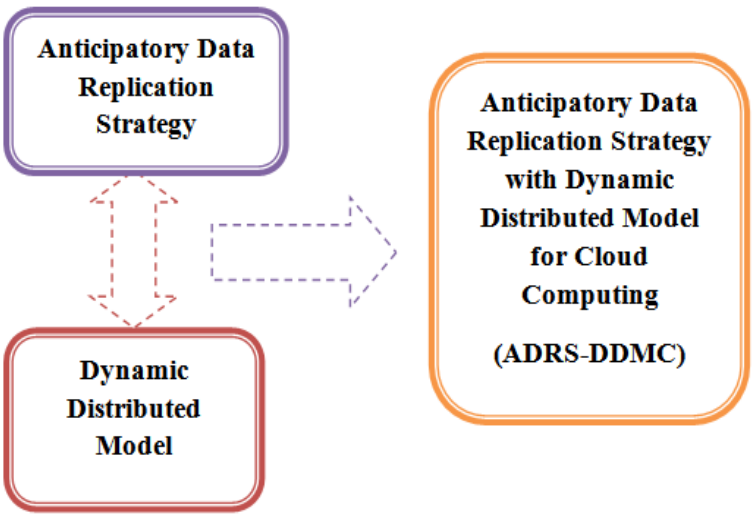

Figure 5.3 Proposed ADRS-DDMC Model

5.3.1 Dynamic Distributed Architecture- In the distributed type of system one, the dynamic load balancing algorithms executed by all nodes present in the system and the task of load balancing is shared among them. By the communication and data exchange in between computing nodes can help achieve proper load balancing. It can take two forms: cooperative and non-cooperative. In the 1st method, the nodes work side by side to achieve a common objective, for example, to improve the overall response time, etc. In the second form, each node works independently toward a goal local to it. The proposed ADRS-DDMC method is based on dynamic distributed architectural design, to improve the response time of a local task. Figure 5.3.1 shows the architectural design of dynamic distributed ADRS-DDMC.

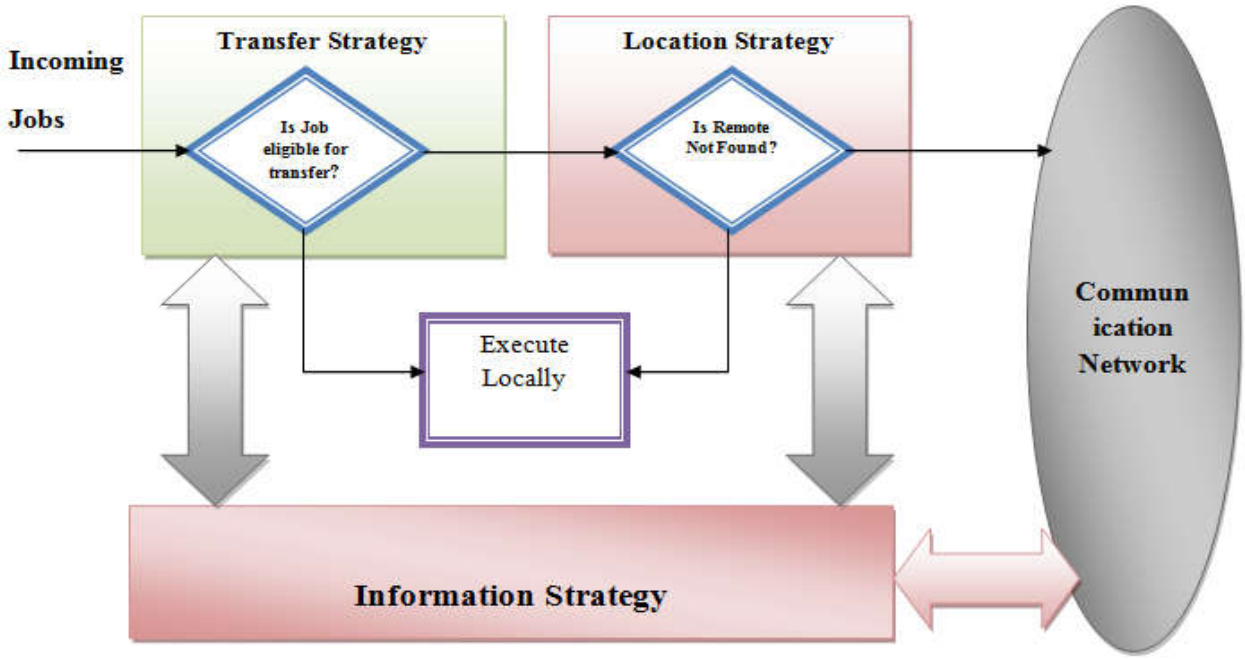

Figure 5.3.1 Dynamic Distributed Model for ADRS-DDMC

A load balancing method which is based on dynamic and distributed nature usually generates more messages as compared to the non-distributed load balancing methods, because in these types of the method each of the computing can directly interact with another node in the system (Chao-Tung Yang, et al., 2011). The main advantages of this approach are it provides better reliability and performance, even if any of the nodes fail or down into the system and it will not affect the entire total load balancing process of the system. Proposed dynamic distributed load balancing method can dynamically manage system load and it also provides a proper load distribution between various nodes.

5.3.1.1 Policies or Strategies in Dynamic Distributed Model of ADRS-DDMC-There are four policies used in the proposed model:

1) **Transfer Policy-**A transfer policy or transfer strategy is a standard in the cloud system which selects a job for or instruction, transferring from one location to another (from a local node to a remote node).
2) **Location Policy-**It describes the policy, which destination node should be chosen for a transferred task.
3) **Information Policy-**In the dynamic load balancing method Information policies are responsible for the collection of information from different nodes in the system.
4) **Selection Policy-**It species the processor involved in the load exchange (processor matching**).**

5.3.2 Data Replication Architecture-Figure 5.3.2 illustrates the high-level architecture of the proposed replication system.

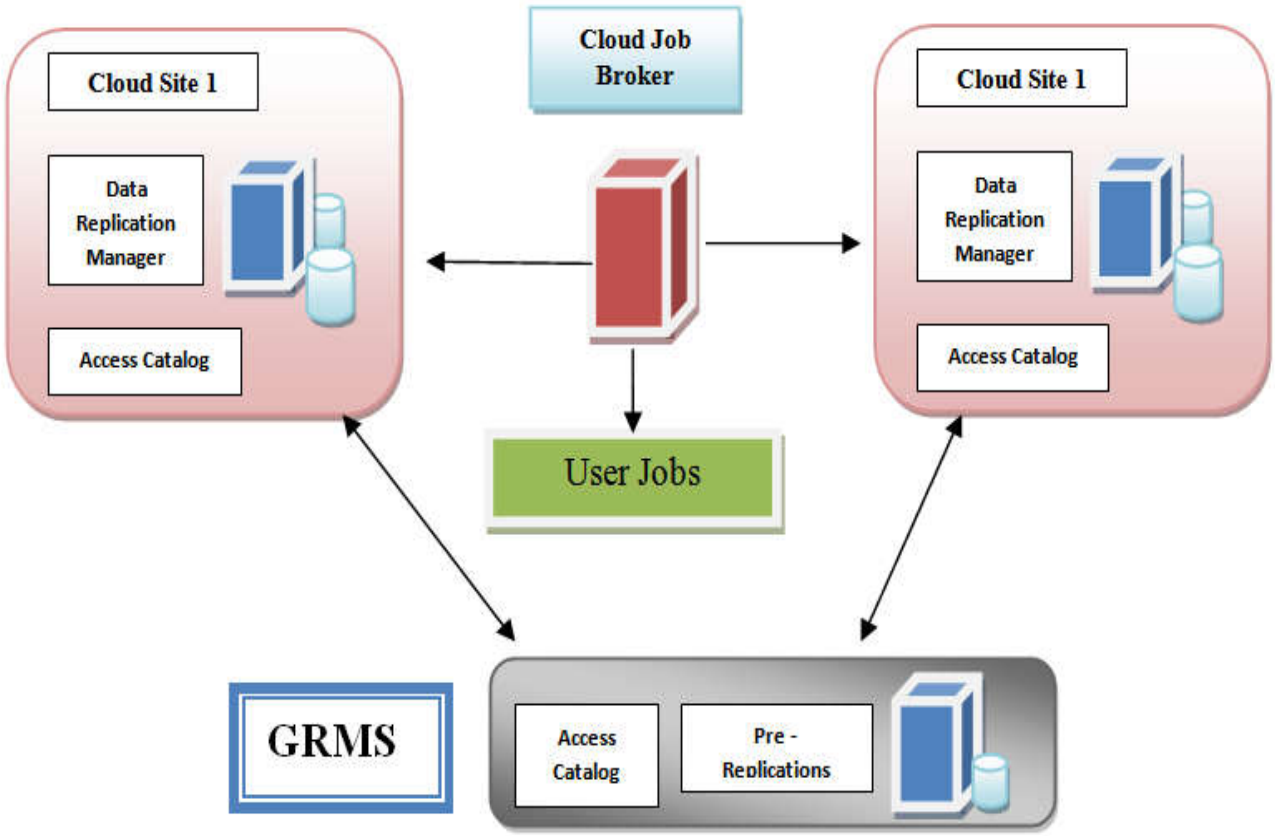

Figure 5.3.2 Proposed ADRS-DDMC Data replication Strategy Design

In this architecture (figure 5.3.2) according to the optimal network transmission rate, job broker will allocate the requested jobs on the first available site. Each site consists of data computing or storage node which is responsible for performing the job and storing all the required data. Also within each site, there is a replication manager component that is managing the files access histories and it defines whether the requested tasks can be completed locally or remotely. Additionally, all sites are connected to the Global Replica Management System (GRMS) and it creates the replicas if required.

Global Replica Management System GRMS consists of two main components:

1) **Global Data Catalog-** It stores the global access patterns.
2) **The Pre-Replication Engine-** It anticipates the replication needs of the sites.

If any communication sites don't have the required files to complete the task, they directly sent a request signal to GRMS and in response, GRMS provides them required replicas. Then in next step, if beneficial, GRMS will send the replicas along with their adjacent files for future access to the site. The main novelty of the proposed algorithm could be highlighted by GRMS component. As described above, GRMS is responsible for creating and replacing the replicas and pre-replicas. Figure 5.3.2 explains the internal architecture of GRMS.

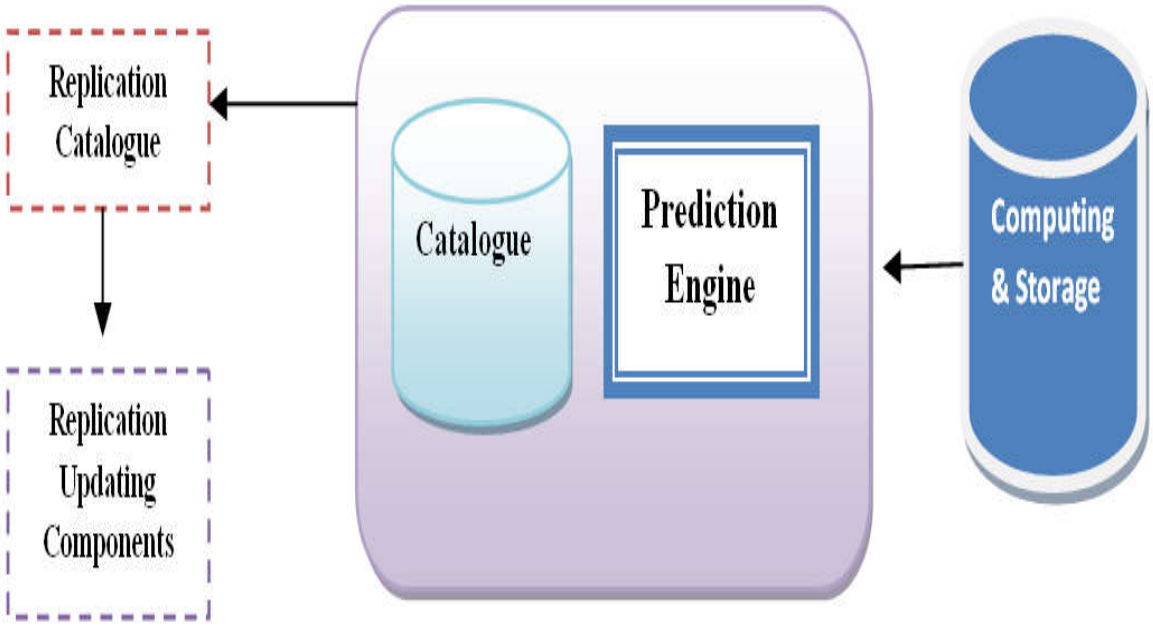

Figure 5.3.2 - GRMS High-Level Architecture

The main elements of GRMS are:

1) **Computing or Storage Node-** Computing element is responsible for running the submitted task which stored in the queue. When sites request a file that is not stored locally, a copy of a requested file will be stored in the storage area.
2) **Pre-Replica Creation Engine (PRCE)-**The PRCE is responsible for managing and creating the pre-replication data that may be requested by sites in future. In this block, prediction engine would access the data catalog to find out the data access patterns. It would then anticipate the data with high access probability.
3) **Replica Catalog-**When replica engine found the data that should be pre-replicated; it stores the name and physical location of that replica in the replication catalog.
4) **Replica Updating Component-**It is responsible for creating the actual replicas. By accessing the replica catalog, this component will select the best replica with minimum communication cost.

5.3.3 Performance Parameters for Proposed ADRS-DDMC Method-In Proposed ADRS-DDMC method the reliable VMs are identified based on:

Parameters of Proposed ADRS-DDMC Model

Method	Performance Parameter
Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing (ADRS-DDMC)	➢ Previous History ➢ Mean Job Execution Time ➢ Effective Network usage ➢ Total Number of Replicas ➢ Memory Availability ➢ Time Consumption

Table 5.3.3 Parameters of Proposed ADRS-DDMC Model

A. **Time Consumption-**The time consumed is calculated in milliseconds. The time bound is granted as a lower bound time limit and the upper bound time limit. In this Process, the response time limit for each VM is given in milliseconds. If the VM can respond within

the designated time limit, then that VM is taken as true. If $R < t_{limit}$, then the request arrives immediately without any waiting time. So, the probability of the timeout and overflow failures do not happen (i.e. the request stage is reliable).

$$\mathbf{R_{request} = E^{n}_{i=1} \int_{1}^{1.5} (t_{limit}) <= 1.5} \qquad \textbf{---------5.3.3.1}$$

The request R will be taken as reliable at the condition t_{limit}

B. Memory Availability-The memory available for each VM is taken separately. If two or more VMs respond to the same time limit, then the reliability is taken based on the memory availability of the VM. The memory availability checker is used to find the best memory utilization VM and that VM is considered as reliable. The resource availability is taken at the time of more than two VMs responding within the time and the same time.

$$\mathbf{M_{min} = \sum_{Vm=1}^{n} \int_{1}^{1.5} (t_{limit}) <= 1.5} \qquad \textbf{---------5.3.3.2}$$

If the waiting time is longer than the due time t_{limit} timeout failure occurs.

C. **The previous History-**In the cloud, the previous reliable VM details are stored in the server database. At the time of processing, all VMs responses exceeding the time set the proposed algorithm goes to the previous story. The reliable VM is taken according to the number of times the previous reliable VM stored in the database with minimum count is taken as reliable VM. For example, VM 1 has been stored more times in the database than the other VMs then VM 1 is taken as true.

The previous history is also a repository area to hold the checkpoints. At the end of each computing cycle, a decision mechanism makes checkpoint in it. In the case of a complete failure, the backward recovery is performed with the help of checkpoints to identify the reliable VM already in the previous history.

The previous history scheme provides an automatic forward recovery. If a node fails to produce an output within time or over time the process will not fail. It will continue to operate with the remaining nodes. This mechanism will produce output until all the nodes

fail. The previous history produces the result of the maximum count of VMs in the previous computing cycle. The previous history gives backward recovery method with high reliability. The backward recover gives increase reliability to identify the reliable VM in the proposed method.

5.3.4 Working of the Proposed ADRS-DDMC Algorithm for Data Replication

The previous section depicted the high-level architecture of the system. In this section, the details of the proposed algorithm will be explained.

ADRS-DDMC algorithm is constructed on the basis of the following assumptions-

- For predicting the future replication needs, past sequence of access patterns should be available in the data catalog.
- The threshold for $\mathbf{T}_{\mathbf{\_Submitted\_File}}$ and, $T_{\_Pointer}$ is set to 50.
- The threshold for the maximum number of accessing the file is 20.

The detail description of the proposed ADRS-DDMC algorithm is provided below:

1. **Job broker**-When jobs are submitted to the system, the job broker will allocate them to available sites.
2. **Computing storage manager (CSM)-**Each site consists of a CSM which is responsible for analyzing the required files for completing the jobs. If the computing storage manager finds that, the site has the required files then the job will be complete locally within the site. Otherwise, computing storage manager will send a request to GRMS and ask for the required replica.
3. **When GRMS receives the replication request-**Firstly will start looking into the stored catalog to check the site's access patterns. Otherwise, computing storage manager will send a request to GRMS and ask for the required replica. When GRMS receives the replication request, it will start looking into the stored catalog to check the sites access patterns.
4. **Storing of access pattern-**Here we have applied the same tier architecture proposed in (Jing Xiao, et al., 2012).
5. **GRMS as a root-**This is noted as layer two of the tier. The tier will be monitors the number of the sites and will store their names.

6. **Stores the existing files located on each site-**This is noted as layer three of the tier. Filename, the latest time that the file has been requested and a number of times that the file has been accessed will be stored in each leaf of the tier. Additionally, each site has a pointer that indicates to the last file that has been accessed on the site.
7. **When sites accessing their replica-**GRMS calculate the last time that the file has been requested.

   If $[T_{Submitted\_File} - T_{pointer}] > Threshold$, --------5.3.3.3

   5.3.3.3 Shows that a new sequence line should be added under the site's name.

   Else if $[T_{Submitted\_File} - T_{pointer}] < Threshold$, ---------5.3.3.4

   It would be considered as a successive file and the child will be added to the end of the last sequence.
8. **When computing storage component of the site notifies GRMS to find a replica-**GRMS will start looking at the access catalog to find a required file. GRMS should search the catalog to find the required file. Also, it should consider the communication cost between the site that has the file and the site that needs the file.
9. **Efficient A* search algorithm-**Therefore to find out which site has the file, an efficient A*, searching method is used to find the shortest path with minimum communication cost.

   A* is a popular searching algorithm for finding the best-first shortest path which satisfies evaluation function. A* search algorithm will use heuristic approaches to avoid the path that has more expensive costs. In A* the evaluation function will be-

   F (n) =G (n) +H (n) ---------5.3.3.5

   G (n) explains the cost of finding the file. H (n) describes the heuristically estimated c (cost) from the source to the destination.

   Hence, A* algorithm will find the target file by considering the transfer cost of the required file shown in equation 5.3.3.6.

   Communication cost=Size of the replica / Bandwidth between servers ---------5.3.3.6
10. **A* algorithm will retrieve the file with minimum communication cost-**If A* couldn't find any files that satisfy the condition it will return 0. Then GRMS will notify the site that the replication is not beneficial and job completion should be done remotely with the site that has the file.

11. **The anticipatory behavior of the algorithm-**To satisfy the anticipatory behavior of the algorithm, GRMS will store the replica name and will look for the children of the file.

    If (Number of accessing the file) > threshold ---------5.3.3.7

    Then it will add the child for pre-replication.

12. **Depending on the location of the replica and based on the business rule that is designed for this algorithm-**GRMS will search three tiers after the replicas location and will select the file with the highest access pattern for pre-replication.
13. **If the required replica doesn't have any child it will retrieve 0 and will exit the algorithm-**Then in the last step of the algorithm, GRMS will start transmitting the replicas and it's adjacent to the site that requested the files.
14. **Replacement procedure by MRU method-**In this phase the replacement procedure will start by using Most Recently Used (MRU) method.

    The total computing storage of the file will check if it has enough capacity for storing the received files. Otherwise, it will check the first stored replicas.

    If $_{t+1}$ (current time) - t_i (last access time) > threshold ---------5.3.3.8

    Then it will remove the old replica and insert the new replica received from GRMS. If still there wasn't enough space for storing the new replica the computing storage will continue removing the old replicas until there are enough spaces for the new replicas.

5.4 PROPOSED ADRC-DDMC METHOD FOR CLOUD

5.4.1 Proposed ADRS-DDMC algorithm for data replication-Following steps are used in the proposed method:

Step-1 GRMS Store the access patterns as

(Requested file name, total number of accessing file and total requested)

Step-2 User request for a file or data

Step-3 Search the history catalog

3.1 If the requested file exists in local server

Then

3.2 retrieve the file and exit algorithm

Else

3.3 A request will be sent to GRMS

Step-4 (Call A* algorithm () for search)

4.1 A* search will be initiated in catalog history to find the physical location of the file between all the available files,

4.2 Calculate the communication cost and select the file with minimum value

Communication cost = Size of the replica / Bandwidth between servers

Step-5 For the selected replica

5.1 Check if it is beneficial to pre-replicate its adjacent files

5.2 Check if the file has hierarchy

then

5.3 If there exists only one child

then

5.4 replicate and exist

else

5.5 For 3 tiers after the replica

5.5.1 Select the child with maximum access number

5.5 retrieve selected replica & it's adjacent

5.4.2 Most Recently Used Replacement Algorithm

Following steps are used in MRU method

Step-1 (Check for new replica files)

1.1 For each new received replica

1.2 If the new received replicated file size < available storage in target server

1.3 Then

1.4 Insert the first replica

1.5 Else

Step-2 (Check for deletion of file)

2.1 Calculate the difference between current time and last time that file has been accessed

2.2 Select the file with minimum access number

2.3 Delete the file

Step-3 Go to step 1

5.5 SIMULATION AND RESULTS

To evaluate the performance of our proposed algorithm Java programming has been used to extend the Cloud-Sim simulation tool. Cloud-Sim is an open source simulation package developed in Java language by the University of Melbourne.

It is mainly developed to study the effectiveness of different optimization algorithm in cloud computing. In Cloud-Sim, we have several sites containing several virtual machines and storage elements. The broker is responsible for load balancing the requests on available resources.

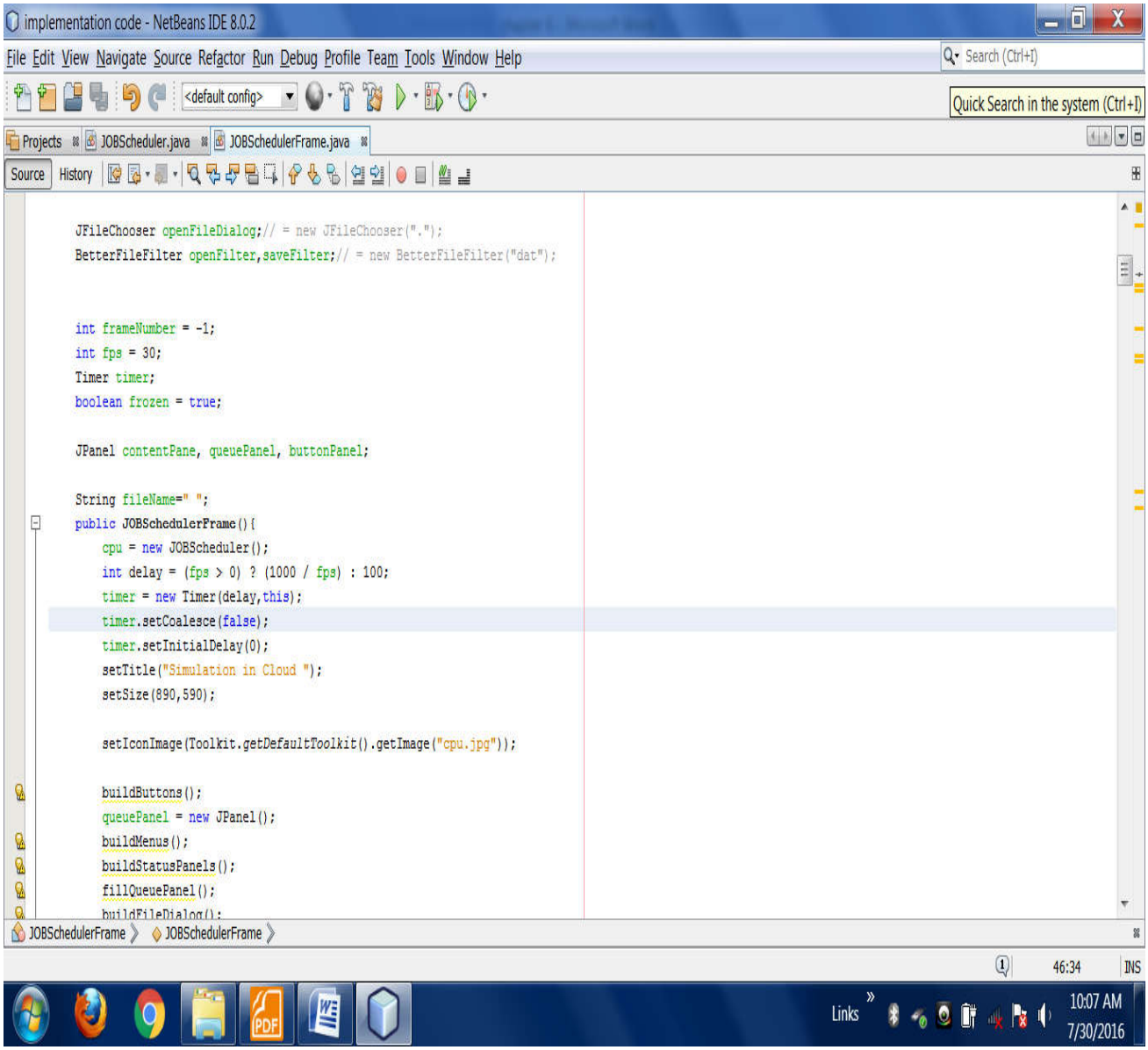

Figure 5.5(a) Creation of VMs for Proposed Method

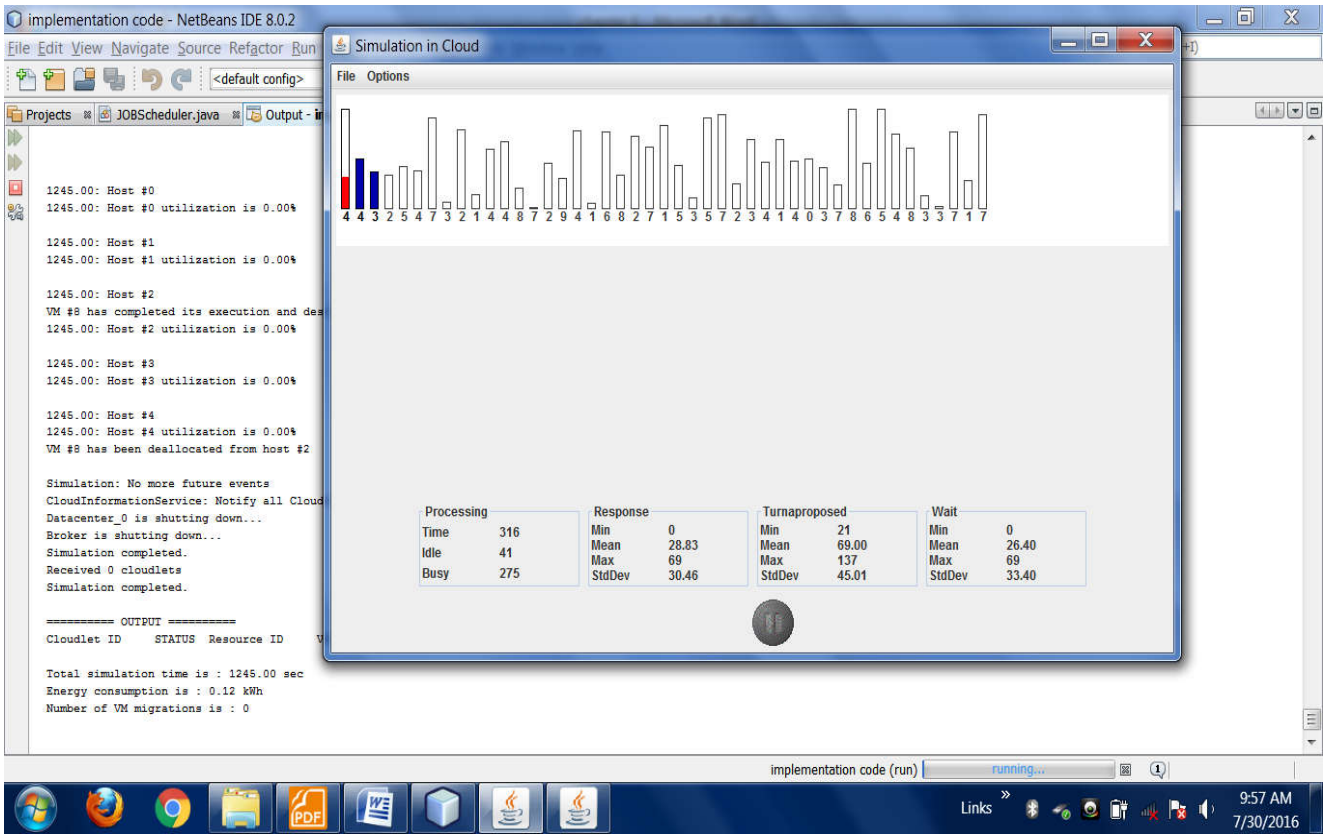

Figure 5.5 (b) Simulation at Runtime for Proposed Method

Each site has a replica manager that handles the automatic replica creation and deletion. Different jobs could be submitted to the broker to be assigned to available resources. The order in which the tasks should be assigned to available resources is determined by the following four main access patterns-

1. **Sequential-** In this access pattern the files are considered as successive request and will be assigned in order.
2. **Random-**Files are accessed randomly.
3. **Unitary Random Walk-**By random direction, the files are selected in a way that the successive files are exactly one element away from the previous file.
4. **Gaussian Random Walk-**Similar to unitary random walk with a difference that the files are selected in a Gaussian distribution.

In order to evaluate the results, Proposed ADRS-DDMC has been compared with existing LRU and LFU replication algorithms. These algorithms are replicating the files based on deleting the

least recently used or least frequently used files. The algorithms have been tested in four patterns sequential, random walk, random access and random Gaussian access. The experiment has been simulated by deploying 3 data centers, with five sites. Each site contains five VMs with 100 jobs. The minimum bandwidths between VMs are 45 Mbit/s and maximum bandwidths between sites are 10000 Mbit/s with total 10 rounds of experiments. These inputs have been hardcoded in Cloud-Sim and will be created at runtime.

The performance evaluation metric that has been applied in the simulations results are Mean job execution time, effective network usage and the total number of replications, time reliability, memory reliability and previous history, which are described in details in next section.

5.5.1 Mean Job Execution Time-Mean job execution time is one of the important evaluation factors. Total execution of all jobs in milliseconds divided by a number of the jobs would highlight the mean job execution. To compare the performance of our algorithm, the mean job has been compared with the existing LRU and LFU algorithm, for 100 jobs. The comparison result is shown in figure 5.5. The simulation results show that our proposed algorithm has the lowest value in minimum job completion. As the algorithm has the functionality to Anticipate the replicas, most of the files can get accessed locally. Ultimately the access time and completion time would be minimized.

Mean Job Execution Time for LRU, LFU and ADRS-DDMC

Access Pattern	Mean Job Execution Time (in Seconds)		
	LRU	LFU	ADRS-DDMC
Sequential Access	1500	1500	1200
Random Access	1200	1100	850
Random Walk Unitary Access	700	500	500
Random walk Gaussian Access	1300	1200	800

Table 5.5.1 Mean Job Execution Time for LRU, LFU and ADRS-DDMC

Figure 5.5.1 Mean Job Execution Time for LRU, LFU and ADRS-DDMC

5.5.2 Effective Network Usage- The metric indicates the ratio of the files that were transferred to the requested site. Lower network usage by a method shows better performances.

ENU = (N file remote access + N file replication) / (N remote file access + N local file access) -------5.5.2.1

The ENU of proposed algorithm has been compared with different algorithms LRU, LFU and ADRS-DDMC in different four patterns such as sequential access, random access.

Access Pattern	Effective Network Usage %		
	LRU	LFU	ADRS-DDMC
Sequential Access	45%	37.50%	32.50%
Random Access	30%	25%	20%
Random Walk Unitary access	20%	14.50%	10%
Random walk Gaussian access	40%	35%	27.50%

Table 5.5.2 Effective Networks Usage % for LRU, LFU and ADRS-DDMC

Table 5.5.2 shows that the proposed ADRS-DDMC algorithm has the lowest value in most cases which is a good indicator of the algorithm efficiency. And that's because by pre-replicating the high probable files; most of the files would be accessible locally and are available at the time of need.

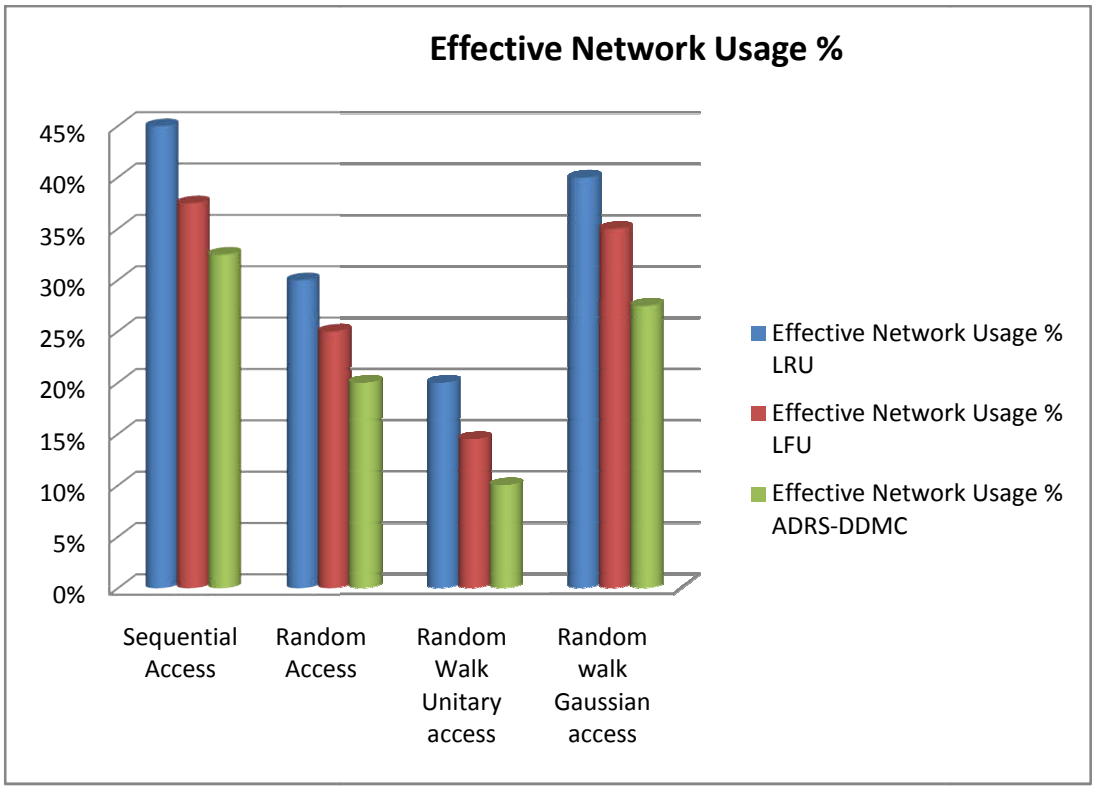

Figure 5.5.2 Effective Networks Usage % for LRU, LFU and ADRS-DDMC

The above results show effective network usages %. In fact, the higher availability of data will decrease the data replication. Therefore, as replication will not happen again, effective network usage will be decreased which has a great impact on load balancing. It should be mentioned that wrong prediction and wrong pre-replicating the files will increase the ENU and consequently it will not have any benefits for the target site rather than consuming more bandwidth.

5.5.3 Total Number of the Replicas-Greater values of the replication numbers indicates that the files were not stored locally and replication procedures were needed to make the files available. Results are calculated for four different access pattern for existing LRU, LFU methods and proposed ADRS-DDMC method. Following results show the performance comparisons of existing and proposed methods.

Total Number of replications LRU, LFU and ADRS-DDMC

Access Pattern	Total Number of Replications		
	LRU	LFU	ADRS-DDMC
Sequential Access	580	500	280
Random Access	410	390	210
Random Walk Unitary access	175	185	125
Random walk Gaussian access	385	400	200

Table 5.5.3 Total Number of replications LRU, LFU and ADRS-DDMC

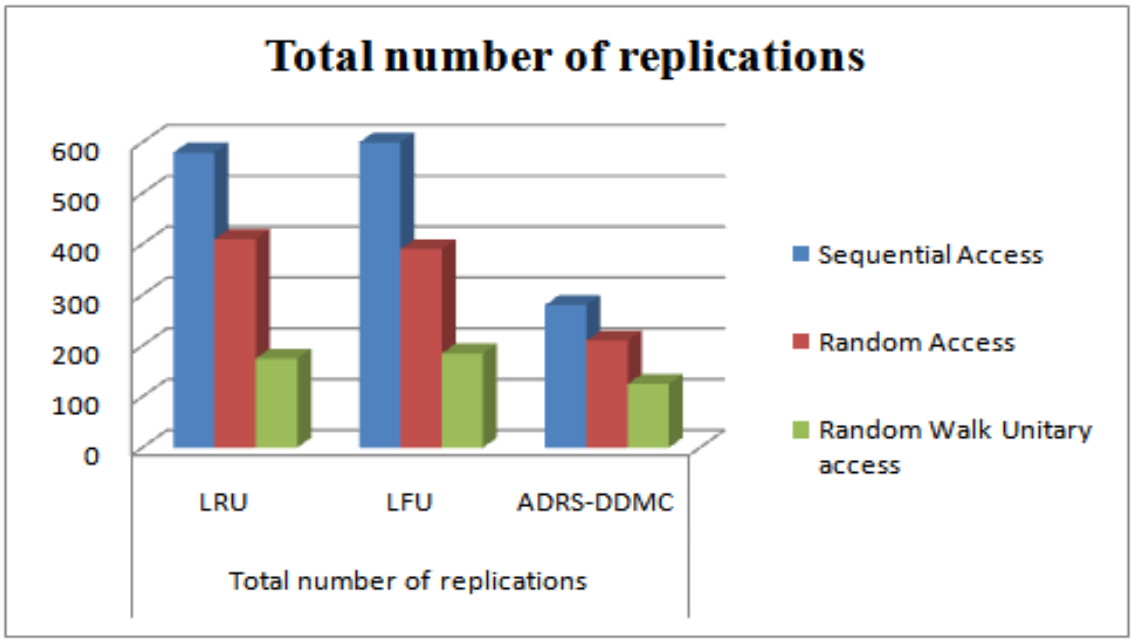

Figure 5.5.3 Total Number of Replications LRU, LFU and ADRS-DDMC

As it is obvious in figure 5.5.3, proposed algorithm ADRS-DDMC provides the lowest number of the replications as compared with LRU and LFU. It predicts the future needs of the network and estimates the files that needed to be locally accessible. Therefore, as the files are pre-replicated before they have been actually requested, at the time of the request the files would be locally accessible and no replication is needed. As a result, total replication number of the files would be decreased.

5.5.4 Time Reliability-The time reliability VM takes the VMs that responds within the time as reliable. The time bound is applied based on the CPU usage. The CPU utilization contains memory usage, latency, execution and speed of each VM (table 5.5.4).

Time Reliability Results for Existing and Proposed Method

Virtual Machine	VM-1			VM-2			VM-3			VM-4		
Cycle	Time in seconds			Time in seconds			Time in seconds			Time in seconds		
	LRU	LFU	ADRS-DDMC	LRU	LFU	ADRS-DDMC	LRU	LFU	ADRS-DDMC	LRU	LFU	ADRS-DDMC
Cycle 1	1.525	1.502	1.245	1.668	1.445	1.335	1.347	1.224	1.025	1.589	1.665	1.125
Cycle 2	1.421	1.398	1.348	1.988	1.345	1.245	1.895	1.114	1.445	1.498	1.488	1.245
Cycle 3	1.478	1.245	1.114	1.745	1.225	1.114	1.457	1.402	1.442	1.389	1.212	1.124
Cycle 4	1.587	1.478	1.224	1.458	1.598	1.445	1.854	1.689	1.335	1.501	1.441	1.355
Cycle 5	1.463	1.335	1.354	1.989	1.755	1.224	1.654	1.544	1.221	1.401	1.331	1.388
Cycle 6	1.235	1.455	1.236	1.645	1.456	1.123	1.445	1.388	1.332	1.33	1.756	1.445
Cycle 7	1.025	1.379	1.112	1.554	1.335	1.778	1.225	1.425	1.098	1.145	1.179	1.222
Cycle 8	1.478	1.456	1.211	1.256	1.545	1.112	1.745	1.655	1.221	1.388	1.426	1.245
Cycle 9	1.356	1.255	1.124	1.334	1.445	1.015	1.456	1.554	1.652	1.401	1.015	1.124
Cycle 10	1.568	1.245	1.112	1.897	1.226	1.113	1.224	1.334	1.221	1.668	1.215	1.102
Average Time	1.413	1.375	1.20	1.6534	1.4375	1.2504	1.5302	1.4329	1.2992	1.431	1.372	1.2375

Table 5.5.4 Time Reliability Results for Existing and Proposed Method

Table 5.5.4 presents the results of 10 computing cycles to identify the reliability among four virtual machines. In the proposed methodology the time is given as 1.0 to 2.0 seconds. The authenticity mechanism identifies the active and failure VMs. The proposed algorithm first checks the status of the VM. The position of all the VMs is sent to the intermediate server. If any VMs fail, the data regarding the failure VMs such as the failure message indication is transmitted to the host. The failure VMs is taken as false and the active VM is taken as true in the VM status list. The reconciliation computation gives the status of responses obtained by the intermediate server within the time limit 1.0 to 2.0 seconds.

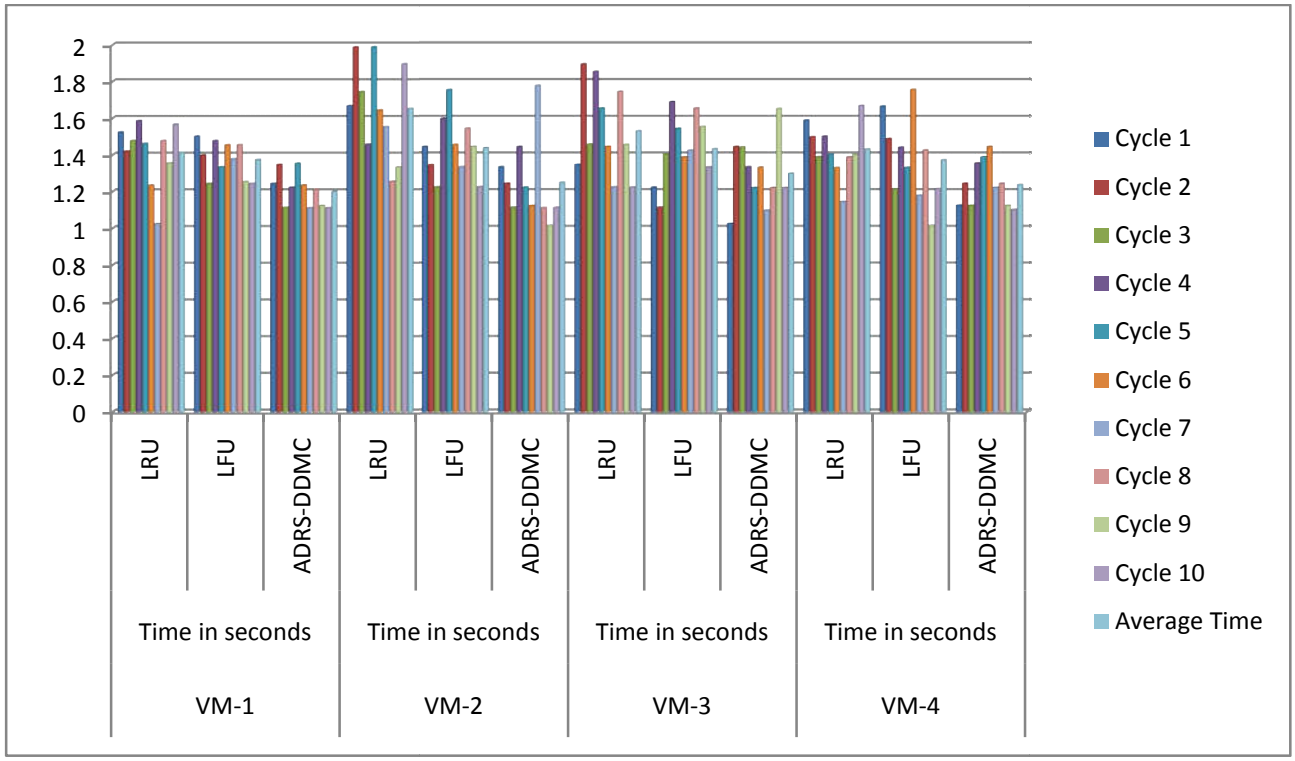

Figure 5.5.4 Time Reliability Results for Existing and Proposed Method

5.5.5 Memory Reliability-The memory utilization is an efficiency parameter. The memory utilization is considered if more than two VM responds within the same time. The VM with less memory usage is considered as the most reliable. This gives efficient results for further storage of the client data request. The reliability is selected based on the less memory utilization and increased speed of VMs. The memory is increased or decreased for different simulation results with different computing cycles the memory utilization not only considers the memory usage but

also take the speed of each VM. The increase in speed is based on the memory usage. The memory usage level is efficient in the proposed method.

Memory Utilization % Results for Existing and Proposed method

Virtual Machine	VM-1			VM-2			VM-3			VM-4		
Cycle	Memory Utilization %			Memory Utilization %			Memory Utilization %			Memory Utilization %		
	LRU	LFU	ADRS-DDMC	LRU	LFU	ADRS-DDMC	LRU	LFU	ADRS-DDMC	LRU	LFU	ADRS-DDMC
Cycle 1	58.7	44.12	41.22	49.5	44.25	37.88	48.77	44.23	39.88	51.25	46.44	31.44
Cycle 2	58.99	43.33	40.12	54.55	43.55	36.55	53.44	41.25	38.66	55.44	44.22	36.44
Cycle 3	55.78	54.77	45.66	55.66	41.23	39.44	56.45	39.77	40.12	57.88	40.22	38.45
Cycle 4	45.66	52.14	44.12	58.9	46.56	40.11	57.88	45.66	44.55	59.66	41.22	33.66
Cycle 5	47.55	46.88	40.55	54.66	40.22	38.77	53.22	41.22	46.55	54.22	38.99	38.99
Cycle 6	41.25	59.88	37.88	52.33	48.99	34.66	51.44	47.88	47.44	51.25	45.66	37.44
Cycle 7	51.33	41.21	44.65	51.44	47.85	36.56	58.99	46.58	40.12	50.33	51.22	34.56
Cycle 8	58.77	40.22	41.25	58.97	46.55	33.21	60.12	47.88	36.55	55.77	41.22	37.51
Cycle 9	50.44	48.9	44.9	50.22	48.9	37.88	56.33	49.55	39.88	58.66	40.22	38.99
Cycle 10	43.24	41.66	43.12	49.8	42.33	33.44	51.99	50.25	37.55	50.11	40.12	30.22
Average Time	51.17	47.31	42.34	53.60	45.04	36.85	54.86	45.42	41.13	54.45	42.95	35.77

Table 5.5.5 Memory Utilization % Results for Existing and Proposed method

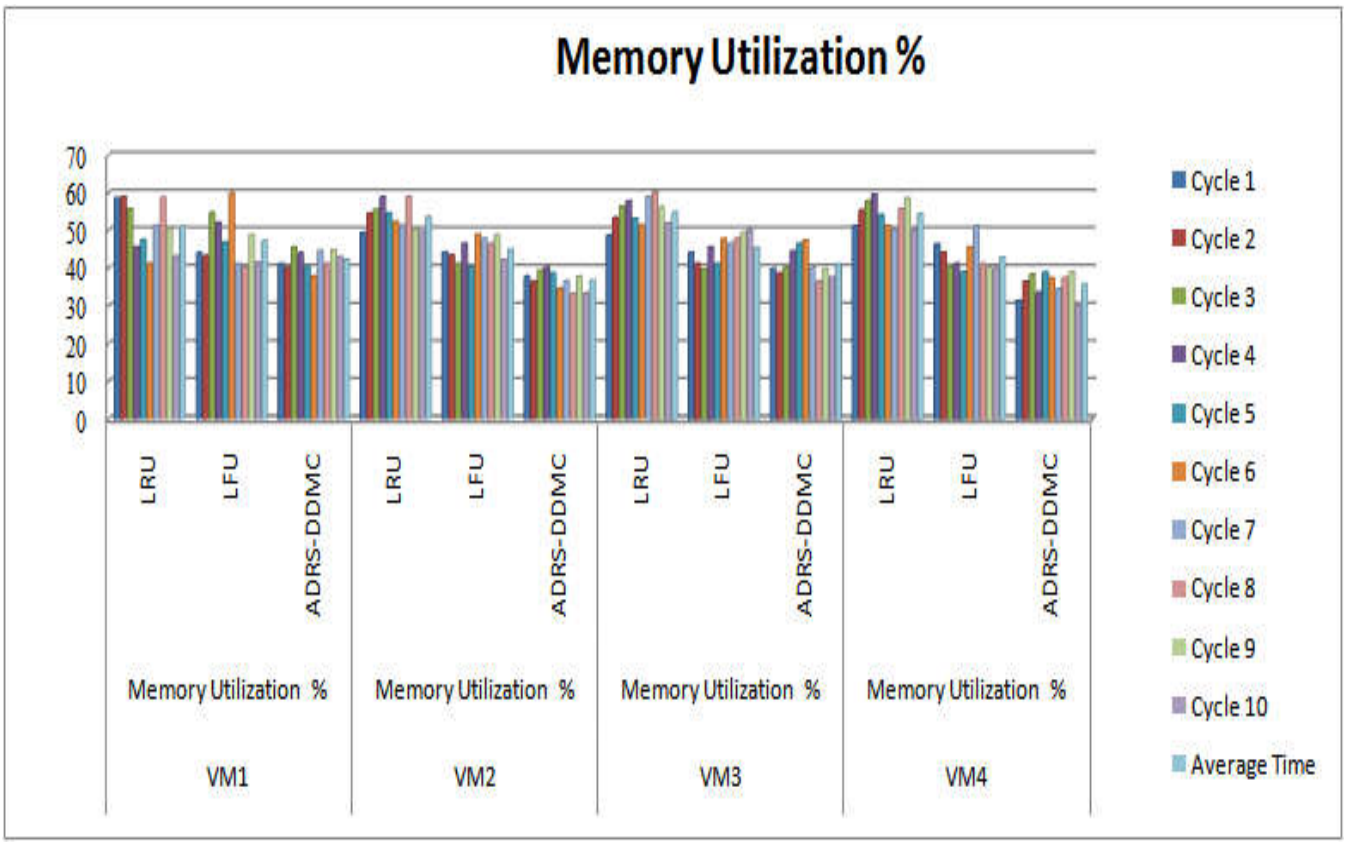

Figure 5.5.5 Memory Reliability Results for Existing and Proposed Method

In above table and figure 5.5.5 clearly, shows that proposed ADRS-DDMC method consumes less average memory % as compared to existing methods for four virtual machines in various cycles.

5.5.6 Previous History Reliability-The previous history is viewed and the reliable VM is selected from the previous computing cycle. The result is stored on the server. The previous history can use the backward recovery. The backward recovery gives an efficient result in the proposed method. In the proposed algorithm the performance of the previous history is to select the maximum count VM to be taken as reliable.

5.6 RESULT ANALYSIS OF PROPOSED AND EXISTING METHODS

Evaluating the results the following points are determined:

a) **Impact of Proposed ADRS-DDMC Method on Mean Job Execution Time-**Proposed ADRS-DDMC estimates the total execution time of the jobs by a mean of 850(seconds)

across four different available patterns (sequential access, random access, random walk unitary access, random walk Gaussian access). Table 5.5.1, compared with LRU and LFU, it can be observed that the total execution time has been improved greatly by the average of 24% and 27% (in seconds) respectively which is a significant improvement.

b) **Impact of Proposed ADRS-DDMC Method on Effective Network Usage-**Proposed ADRS-DDMC improved the network usage by mean of 13.09 % across four different patterns. Table 5.5.3 compared with LRU and LFU it can be observed that the total execution time has been improved incredibly by an average of 32% and 20% (in milliseconds) respectively.

c) **Impact of Proposed ADRS-DDMC Method on Total Number of Replications-**Proposed ADRS-DDMC optimized the total numbers of the replication by a mean of 207.5 across four different available patterns (Sequential access, random access, random walk unitary access, random walk Gaussian access). Table 5.5.4 compared with LRU and LFU, it can be observed that the total execution time has been improved notably by the average of 45% and 47% respectively.

d) **Impact of Proposed ADRS-DDMC Method on Time Reliability-**Proposed ADRS-DDMC improved the time reliability in each cycle. Table 5.5.4 shows the results of the proposed method compared with LRU and LFU; it can be observed that the time reliability has been improved in the proposed method. Proposed method takes less time.

e) **Impact of Proposed ADRS-DDMC Method on Memory Reliability-**Proposed ADRS-DDMC improved the memory reliability in each cycle. Table 5.5.5 shows the results of the proposed method compared with existing LRU and LFU method. An experimental result clearly shows that the memory reliability has been improved in the proposed method. Proposed method takes less time.

CHAPTER 6

ADVANCED ANTICIPATORY PERFORMANCE IMPROVEMENT MODEL, FOR CLOUD

This chapter mainly emphasizes on **Advanced Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC). Proposed AAP-IMC Model** uses a combined strategy of the proposed method of chapter 4 MFL-APSO methods and proposed a method of chapter 5 ADRS-DDMC, to optimize the total execution time of tasks in the workflow applications and load balancing.

6.1 INTRODUCTION

In cloud computing, various parameters such as virtual machines, storage, memory processor and jobs are used as resources. An efficient and optimum utilization of computing resources are always desirable and challenging in cloud computing. In such a scenario load balancing algorithms play an important function where the intention is to schedule the tasks effectively and balance the entire load effectively and can reduce the total response and execution time of the entire cloud system and improve the resource utilization.

This is achieved in proposed methods MFL-APSO (Chapter 4) and ADRS-DDMC (chapter 5). The proposed methods (MFL-APSO and ADRS-DDMC) minimize the total tasks execution time by verifying the load fluctuations of the interconnected tasks and it is based on anticipations to create the pre-replicas for future needs of the sites. The method optimizes load balancing by increasing the data availability among the existing sites. The new era of cloud computing technology will focus on how effectively and efficiently the infrastructures are instantiated and available resources are utilized dynamically.

In the cloud computing resources need to be allocated and their load must be scheduled in a way that provides higher and efficient utilization of computing resource and cloud users can meet their requirements. Cloud computing has several challenges load balancing is one of them. Load balancing methods distribute the dynamic workload across multiple virtual machines (VMs) to

ensure that an entire cloud system, not a single computing resource is either overloaded or underutilized. This can be considered as an optimization problem and an efficient and effective load balancer should use its strategy to the changing environment and the types of tasks. There have been different types of load balancing algorithm are proposed and implemented by various cloud researchers cloud computing.

The main aim of load balancing algorithm is to improve the performance and quality of service and maintaining the efficiency, effectiveness and fairness of the jobs and reduce the execution cost. These existing load balancing methods are encounters with various challenges during dynamic workload distribution for a heterogeneous public cloud in peak time. It can degrade system performance. Once the resources are allocated, effective schedule or load balancers are required, that can allocate jobs more effectively and balanced the entire loads.

6.2 PROPOSED AAP-IMC MODEL FOR CLOUD

In cloud computing, better performance of the system is always desirable and challenging. The performance of the cloud system depends on various factors such as resource availability, resource utilization and distribution, task load balancing and load balancing. Availability plays an important role in Cloud-based systems.

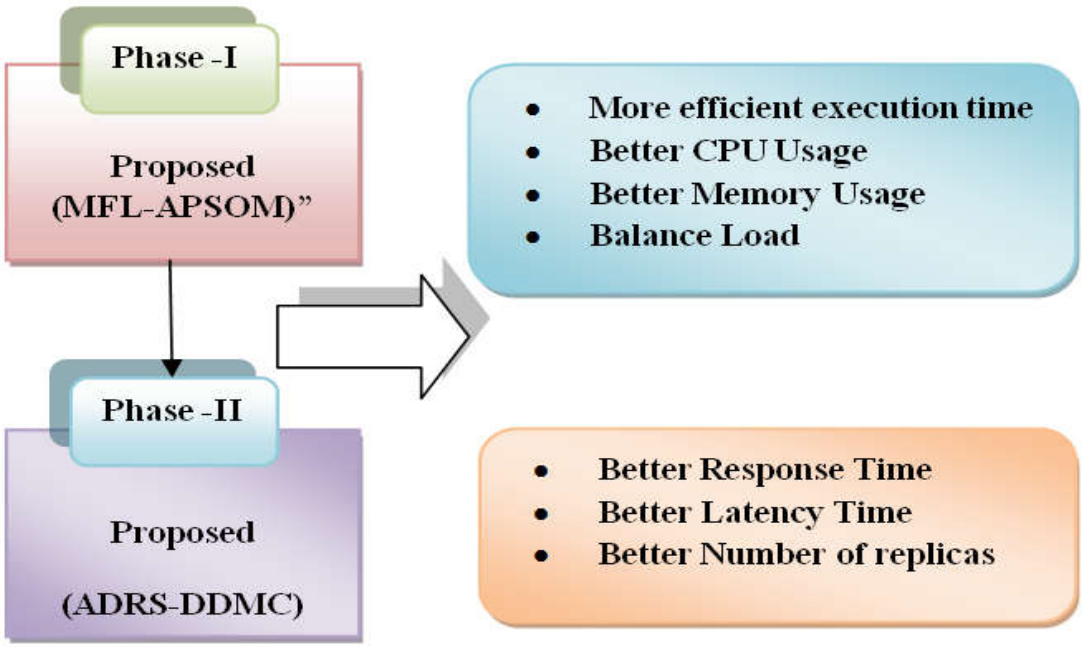

Figure 6.2 Proposed AAP-IMC Architecture for Cloud

Proposed performance improvement model Advanced Anticipatory Performance Improvement Model for Cloud Computing (AAP-IMC) presents a set of solutions for load balancing in the cloud. The proposed model uses a combined strategy of two proposed methods MFL-APSO and ADRS-DDMC in various phases. In cloud computing load balancing method is utilized by various data centers to avoid unavailability of the network. Load balancing achieved through the reduction in software failures and decrease in computer hardware usage task. To optimize the total execution time of tasks and instructions in the various workflow applications proposed method use following phases:

6.2.1 Performance Parameter in Proposed AAP-IMC MODEL-For performance comparison in between proposed and existing method following parameters is used. Each parameter is calculated secretly for both methods (proposed and existing); performances metrics and result comparisons for the cloud load balancing algorithms are based on following factors-

Performance parameter in AAP-IMC Model

Method	Performance Parameter
Advance Anticipatory Performance Improvement Model For Cloud Computing (AAP-IMC)	**Phase 1 MFL-APSOM** ➢ Response Time/ Latency ➢ Resource Utilization ➢ Data Transfer Rate ➢ CPU Load & CPU Time ➢ Memory Rate ➢ Execution Time ➢ Workflow and Soft error rate **Phase 2 ADRS-DDMC** ➢ Time Consumption ➢ Memory Availability ➢ Previous History ➢ Mean Job Execution Time ➢ Effective Network usage ➢ Total Number of Replicas

Table 6.2.1 Performance Parameter in AAP-IMC Model

1. Average Waiting Time-In cloud computing, a waiting time can be defined as how long a computing process has to wait before it gets its time slice. In load balancing algorithms such as First Come First Serve (FCFS) and Shorted Job First (SJF), we can find that waiting time easily when we just queue up the jobs and see how long each one had to wait before it was serviced. When it comes to Round Robin or any other pre-emptive algorithms, we find that a task which has higher execution time for jobs, spend less time in CPU and when they are pre-empted and then wait for some time for its turn to execute or run and at the process starts runs from its same point till completion.

2. Average Response Time-It is the amount of time taken from when a process is submitted until the first response is produced (Kathleen Ericson, et al., 2011). Average response times for each algorithm have decreased by increasing the number of CPUs.

3. Makespan Time-A makespan can be defined as the overall task completion time. We denote completion time of task Ti on VMj as CTij.

$$\textbf{Makes-pan}=\max\{CTij \mid i \in T, i = 1, 2 \ldots n \text{ and } j \in VM, j \in 1, 2 \ldots m\} \quad \textbf{-----6.2.1.1}$$

4. Throughput-It is the total number of tasks or jobs that have completed their execution on a given scale of time. It is required to have high throughput for better performance of the system.

5. Associated Overhead-It describes the amount of overhead during the implementation of the load balancing methods. An associated overhead is movements of tasks, interns process communication and inter-processor. For an efficient load balancing method, minimum overhead is always desirable.

6. **Migration Time-**It is the amount of time for a process to be transferred from one system node to another node for execution. Less migration time shows better performance for the system.

7. Performance-It is the overall efficiency of the system. If all the above parameters are improved and perform their best, then the overall system, performance can be improved.

6.2.2 Design of the Proposed AAP-IMC Model-The proposed AAP-IMC model has a capability to identify free virtual machines earlier and transfer overloaded node to the appropriate free destination. The proposed system is based on following three modules, shown in figure 6.2.2.

Modules in Proposed AAP-IMC Model-Proposed model has following modules:

a. **Load Indicator Module-**The load indicators module indicate current system load and capacity and also capable of periodic all the incoming load. The load indicators module uses two components one is load calculator and second is load updater. Load calculator parameter is used to calculate system load at various levels. Load update parameter updates the system load status into the database.

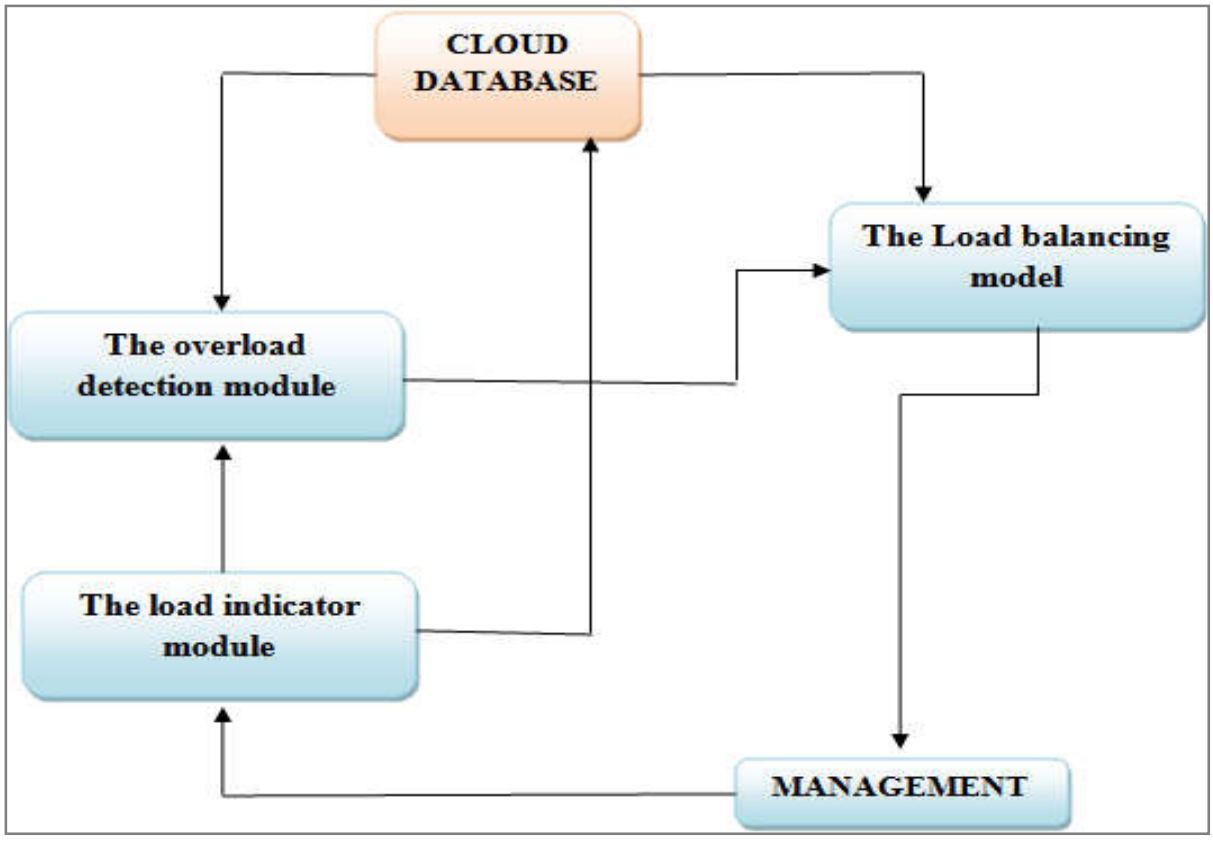

Figure 6.2.2 Design of Proposed System

b. **The Overload Detection Module-**This module identifies the overloaded nodes into the cloud environment. This module uses the information which provided by the load indicator module and system database.

c. **The Load Balancing Model-**This module is the main module of the system. This module is responsible for equal load distribution among all the virtual machines, not a single machine should be overloaded.

6.3 PROPOSED ALGORITHM AAP-IMC MODEL FOR LOAD BALANCING

The proposed methodology AAP-IMC Technique use following steps-

Inputs: Number of Cloud brokers, server, Cloudlets, virtual machines, data centers and number of cloud resources

Output: Load balance will balance more efficiently on cloud machines and generates better throughput

Step-1 Cloud Basic environment variable is created

1.1 Number of Virtual Machines set VM_i (where i=1 to n)

1.2 Number of cloudlets or user set CL_i (where i=1 to n)

1.3 Create cloud Broker B

1.4 Create data center DC_i

1.5 Task represents the set T= $\{T_1, T_2 \ldots\ldots\ldots\ldots T_n\}$

Step-2 Calculate cloud system capacity

2.1 Assign priority for task parameter

T_{high}, $T_{med\ and}$ T_{low}

Where

T_{high} = high, T_{med} = medium and T_{low} = low

2.2 Assign instruction of task T_{high}, $T_{med\ and}$ T_{low} to I_{high}, I_{medium} and I_{low}, respectively to a VM.

2.3 Calculate Capacity CVM of a virtual machine VM-

CVM = (Number of processors in VM) * (Number of instructions of all Processors) * (communication bandwidth ability) -----6.2.2.1

2.4 Calculate Capacity C of all VMs or Capacity of data center

$$C = \sum_{i=1}^{m} C_i \qquad \text{-----6.2.2.2}$$

Step-3 Compute the load degree and calculate the average cloud partition degree from the node load degree

$$\text{3.1 Load degree (N)} = \sum_{i=1}^{m} \alpha_i F_i \qquad \text{-----6.2.2.3}$$

Where $\alpha_i F_i$, ($\sum \alpha_i = 1$), shows weights that represent different values for different kinds of jobs and N represents the current node.

3.2 Calculate the average cloud partition degree from the node load degree -

$$\text{Load}_{\_degreeavg} = \sum_{i=1}^{n} \text{Load degree } (N_i) / n \qquad \text{-----6.2.2.4}$$

Step-4 Calculate processing time of a Virtual machine and all Virtual machines-

4.1 Processing time PT_i, of all Virtual machine

$$\mathbf{P_{Ti} = \text{ Load of all VMs in a data center / Capacity of all VMs}} \qquad \text{-----6.2.2.5}$$

Step-5 User request are assigned to queue

5.1 Number of users from various location send request to cloud for their job processing Jobs J_i

5.2 Each jobs having number of instructions or requests

5.2.1 Calculate computational time (Makespan) of each job

$$\text{Make-span} = \max \{C_{Tij} \mid I \in T\},\ i=1, 2 \ldots n \qquad \text{-----6.2.2.6}$$

Where C_t = finishing time

5.2.2 Check priority for each job

5.2.2.1 If any job has higher priority in queue will take first positioning in queue

Task T_h will take higher position then T_l and T_m

Step-6 Check nodes load status levels

6.1 **Idle When-**

Cloud_Load_degree (N) is set to 0,

And there is no job being processed by this node so the status is changed to Idle.

6.2 **Normal State-**

When $0 < \text{Cloud\_Load\_degree (N)} <= \text{Cloud\_Load\_degree}_{high}$

The node is normal and it can process other jobs.

6.3 **Overloaded**

When- $Cloud\_Load\_degree_{high} <= Cloud\_Load\_degree\ (N)$

The node is not available and cannot receive jobs until it returns to the normal.

Step-7 (Proceed the jobs which are in ready queue)

7.1 Submit the list of tasks T= {T_1, T_2 ……. T_n} by the user.

7.2 Get the available virtual resources from the data center.(for i.e., V_{M1}, V_{M2}…V_{Mn})

7.3 Check if (Standard deviation< Threshold time)

System load is balanced and Exit

$$\textbf{Where } \Omega = [(1/3 * \sum_{i=1}^{m} (PT_i - PT)^2]^{1/2} \qquad \text{----6.2.2.6}$$

m= No of Virtual machine VM

Ω= Standard deviation

P_T=Load / Capacity

T_s=Threshold value

7.4 Processing of queue Jobs

7.4.1 Select each job one by one

7.4.2 Process jobs it into the VMs by considering the assign priority and load

7.5 Compute the fitness value, $\Omega \leq T_s$,

Where threshold value Ts is in between 0 and 1.

7.6 Based on Fitness value- Update the available source position by-

$$Eij = (Xij * W_{ij}) + 2*(L_{ij} - 0.5) * (X_{ij} - X_{kj})\ L1 + Q_{ij(}X_{ij} - X_{kj})\ L_2 \qquad \text{-----6.2.2.7}$$

Where- W_{ij}=L1= 1 / (1+ exp*(- Fitness (i) / A_P))

L_2=1 if process is onlooker one

L_1, L_2 – are fixed number, o or 1

L_2=1 / (1+ exp*(- Fitness (i) / A_P)), if a process is busy one

X_{ij}=nearest neighborhood search solution of working process

W_{ij} =initial weight

X_{kj}=nearest search solution of onlooker process.

A_p= Fitness value in first iteration

L_{ij}=Random numbers between [0, 1] for working process.

Q_{ij}=Random numbers between [0, 1] for onlooker process.

7.7 Working process share his information related to neighborhood position with onlookers process.

Step-8 repeat step 2 to 7, for each iteration unless the best solution is not found

Step-9 after allocating all tasks, check the load of the VMs.

9.1 If any virtual machine VM is overloaded, it goes for next under loaded VM and assigns the task

9.2 After completing each task, repeat the process for all the available jobs/tasks till the system become balanced

6.3.1 Working Steps of Proposed AAP-IMC Model

Proposed method use following steps:

Step-1 The load balancer gets the data from CSP. The load balancer is a middleware intermediate to CSP and data center.

Step-2 The middleware initially sends the request to the data center to identify the failure and reliable VM within the time limit 1 to 1.5 seconds.

2.1 In data center there are multiple VMs are available.

2.2 The data center receives the request from middleware and transfers the request to all VMs in their data center.

2.3 All the VMs respond to the middleware depending upon the status such as total memory, used memory, free memory, latency rate and execution rate are calculated

Step-3 The middleware lists the VM based on the response time except for failure VM. The middleware allocates the load based on the priority. The priority is assigned

3.1 If the VMs respond within the time, the fewer time limits will take first and so on

3.2 If two or more VMs respond at the same time limit, the less memory utilization VM needs as the next priority.

3.3 If the VM exceeds the time limit will assign as last priority and is the two or more VMs respond exceed the time limit the priority is assigned based on the IP address.

Step-4 Resource level percentage has calculated the percentage. This process is done for each VM separately.

Step-5 Using the modified fuzzy rules the resource level is identified as low, medium and high.

Step-6 The load is assigned to the VM based on by with the percentage identified by the resource level percentage parameter.

6.4 EXPERIMENTAL RESULTS FOR PROPOSED AAP-IMC MODEL

The proposed algorithm uses 200 VMs in a single data center. The memory for each VM can share the physical resources on a server. The different type of file size is given and the files allocated based on the resource level percentage taken in allocated VM.

6.4.1 Simulation Parameters-Simulation is done by using Cloud-Sim 3.0 simulator. Following parameters were used for simulation for round robin, existing method and proposed a method. Following results are calculated for each method.

Comparisons Parameters for Simulation

S. No.	Cloud Devices	No. of Used Devices	Physical Characteristics
1	Cloudlets	100-1000	Length = 150000 Bits
			PES number = 1
			File size = 300 MIPS
			Output size = 300 MIPS
2	Brokers	10	NA
3	Data Centers	10-100	Max power = 250
			Static power percent = 0.7
			RAM = 10000
			Storage = 1000000
			BW = 100000
4	Virtual Machines	5-200	Pes number = 1
			RAM =512
			BW = 2500
			Size = 2500

Table 6.4.1 Comparisons Parameters for Simulation

6.4.2 Simulation Snapshots-The Figure 6.4.2 shows the working simulation environment of the proposed methodology.

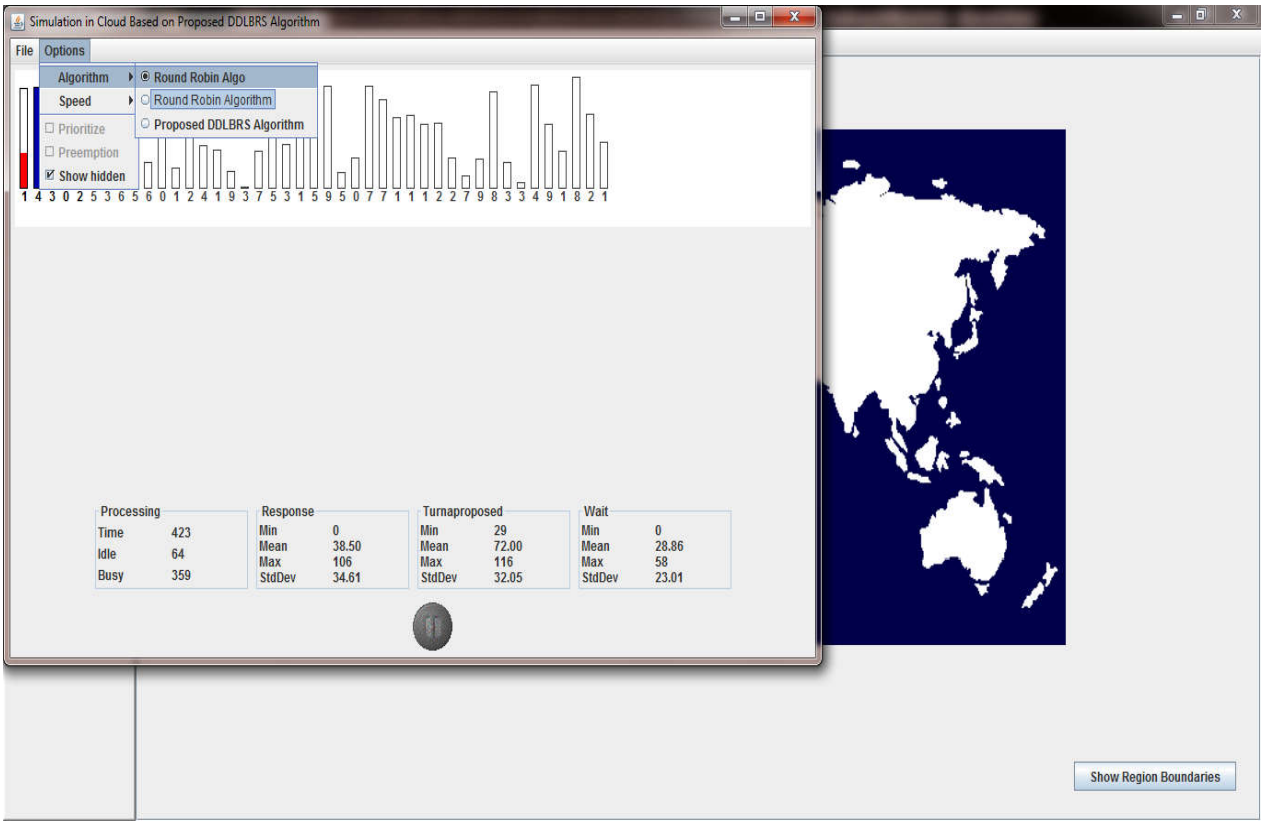

Figure 6.4.2 Initial Simulation of the Proposed and Existing Methods

It contains processing of resources, response time, turnaround time and waiting time for the resources. Following are the comparison parameters proposed over existing. The simulation consists of various load balancing algorithms, which runs on cloud environment for the load balancing of resources in the cloud.

6.5 EXPERIMENTAL RESULTS FOR PROPOSED METHOD

A fuzzy rule is designed in the proposed algorithm. These rules determine the performance of the network and the VMs as low, medium and high. The rules are used to identify the performance of the VM in the cloud. The efficiency is proved based on the performance of the VM such as average response time and average execution time of the VM in a data center. The response time of the VM is reduced and the latency rate is also reduced by using the proposed methodology.

The simulation analysis has taken 200 virtual machines with three processors and different computing cycle. For every computing cycle, a load of each processor in the VM is increased and decreased to get the different results. For each computing cycle, the speed, latency, memory utilization and the response time give better performance. For every computing cycle, the VM is increased gradually and results in better performance. This experiment gives efficient load balancing with the proposed algorithm AAP-IMC. The simulation result is obtained from the parameters latency, response time, execution time and resource level percentage.

During the performance evaluation, the parameters used in the proposed AAP-IMC are CPU utilization rate and the memory usage rate in percentage. By that, we can consider all the decision parameters which can balance and distributes the entire load for the VMs. By considering the performance schedule is generated by balancing the given load. In performance analysis, the maximum utilization of CPU rate of each VM can perform differently in the AAP-IMC algorithm. The analysis gives an account of, how efficiently the proposed approach performs under different load levels of CPU rate. The maximum utilization of the CPU varies for each VM. The memory utilization is calculated as the utilization of the memory in the VM. The percentage of memory is taken in each VM and the load is allocated based on the percentage level. The CPU, memory utilization and processing time are considered as the loads to get the resource level percentage. The load balancing parameter should be designed in such a way that the VM will be selected for a load with the percentage allocated to the VM.

The comparative analysis of the proposed AAP-IMC approach with the existing round robin, honeybees and proposed MFL-APSO, ADRS-DDMC approach gives the expected result. The proposed approach utilizes the CPU rate more efficient than the existing approach under the load balancing condition. So, in the load balanced condition and the CPU utilization rate, the proposed approach has better efficiency over the existing approach. The AAP-IMC helps to determine the overall allocation of resources with the help of the resource level percentage parameter. The proposed methodology completes optimality check which results in reducing the processing and response time. The execution time is to execute any task over the cloud is also reduced.

Following results are calculated for existing methods and proposed techniques:

6.5.1 Data Transmission Rate-Data transfer rate is measured by how much amount of data is transferred in a particular time period. The table 6.5.1 and figure 6.5.1 shows the analysis of the total bits to be transmitted over a certain time of the communication, in between 200 VMs.

Data Transmission Rate %

Number of VMs	Data Transfer Rate %				
	RR Method	Honeybee Method	MFL-APSO	ADRS-DDMC	Proposed AAP-IMC
5	77.5	81.54	84.6	85.2	87.4
10	78.5	81.9	84.9	85.99	88.25
20	79.5	82.1	85.5	86.3	88.9
40	80.44	82.89	86.66	86.9	90.1
50	81.11	83.5	87.2	87.5	90.22
75	81.89	83.9	88.1	88.9	90.44
100	82.55	84.6	89.05	89.5	91.5
125	83.21	85.6	89.6	90.1	91.9
150	83.8	86.66	89.9	90.8	92.5
200	84.1	86	90.1	91.25	93.25

Table 6.5.1 Comparison of Data Transmission Rate for Existing and Proposed

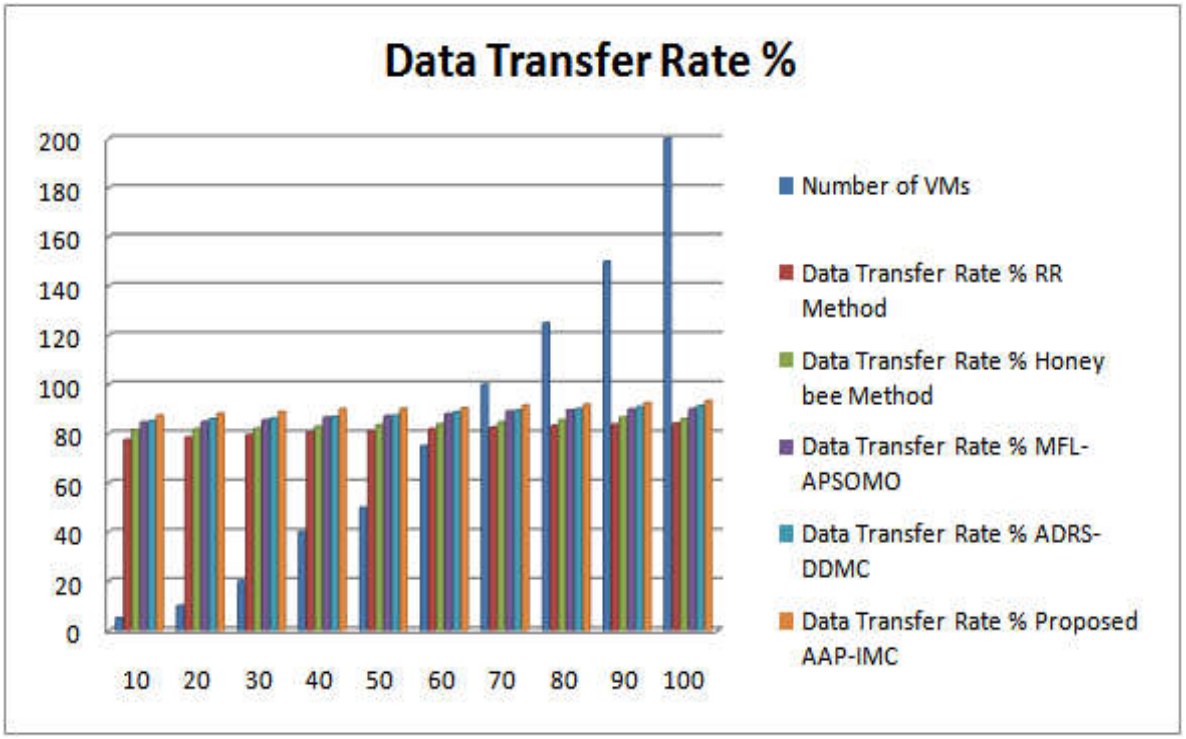

Figure 6.5.1 Comparison of Data Transmission Rate %

Figure 6.5.1 shows results of data transfer rate % for the existing and proposed method. The graph is plotted between the number of VMs (Y-axes) and data transfer rate % (X-axes). Table and figure 6.5.1 reveal the comparison of the data transfer rate % for the 200 VMs and the result shows there is an increase in data transfer rate by using the proposed AAP-IMC algorithm.

6.5.2 Response Time-It is the amount of time taken from when a process is submitted until the first response is produced. Less response time shows better efficient performance.The table 6.5.2 and figure 6.5.2, shown below is the response time.

Average Response Time in ms for Existing and Proposed Method

Virtual Machine VM	Average Response Time (milliseconds)				
	Round Robin Method	Honey Bees Method	MFL-APSO Method	ADRS-DDMC Method	Proposed AAP-IMC
10	37047.72	36147.28	35277.11	35287.79	34567.11
20	35674.74	34678.89	33566.89	33456.02	32456.8
30	27898.45	26457.88	25445.55	24485.66	24478.55
40	23245.55	22140.55	21800.88	20566.66	20155.6
50	16455.54	15455.66	14600.25	14552.55	14301.22
75	12978.33	12878.5	12789.55	12405.55	12304.6
100	10478.56	10302.55	9899.99	9788.56	9605.88
125	9025.44	8998.36	8901.44	8708.9	8608.9
150	8578.77	8478.93	8520.22	8444.32	8309.9
200	7448.02	7389.99	7244.36	7104.33	7080.33

Table 6.5.2 Average Response Time in ms for Existing and Proposed Method

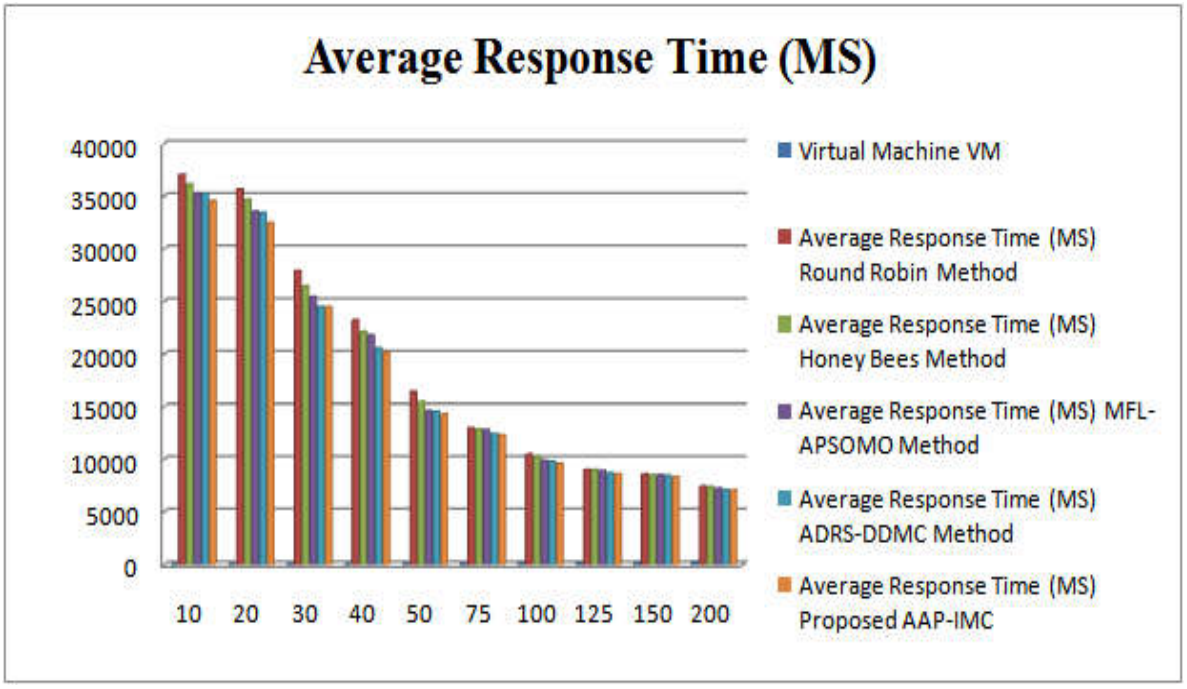

Figure 6.5.2 Average Response Time in ms for Existing and Proposed Method

Figure 6.5.2 shows results of average response time % for the existing and proposed method. The graph is plotted between the number of VMs (X-axes) and average response time in milliseconds (Y-axes). Table 6.5.2 shows the comparative analysis of average response time of the 200 VMs. Above results (table 6.5.2) clearly shows that proposed (AAP-IMC) method shows more efficient results than the existing method.

6.5.3 Execution Time-The execution time is the total amount of time the virtual machine takes in executing the instructions. The time of an executing program therefore is generally much less than the total execution time of the program.

Results for Average Execution Time

Virtual Machine VM	Execution Time (Milliseconds)				
	Round Robin Method	Honey Bees Method	MFL-APSO Method	ADRS-DDMC Method	Proposed AAP-IMC
10	88901.45	77898.66	66745.88	65447.88	62455.55
20	77458.66	70899.89	61455.55	50145.55	49789.65

30	68456.66	61254.33	50145.22	48788.66	48100.33
40	62455.33	55664.55	46452.33	41256.67	40125.26
50	56455.67	46554.55	35667.62	33214.33	33000.99
75	48989.66	37899.74	31245.66	30144.78	29878.89
100	43456.55	33214.98	30777.22	26466.55	25415.68
125	36455.55	24545.69	20155.11	14788.99	14665.33
150	31255.33	20898.32	18788.98	16554.55	15545.6
200	25456.55	14777.87	12111.55	11045.66	10056.68

Table 6.5.3 Average Execution Time in ms for Existing and Proposed Method

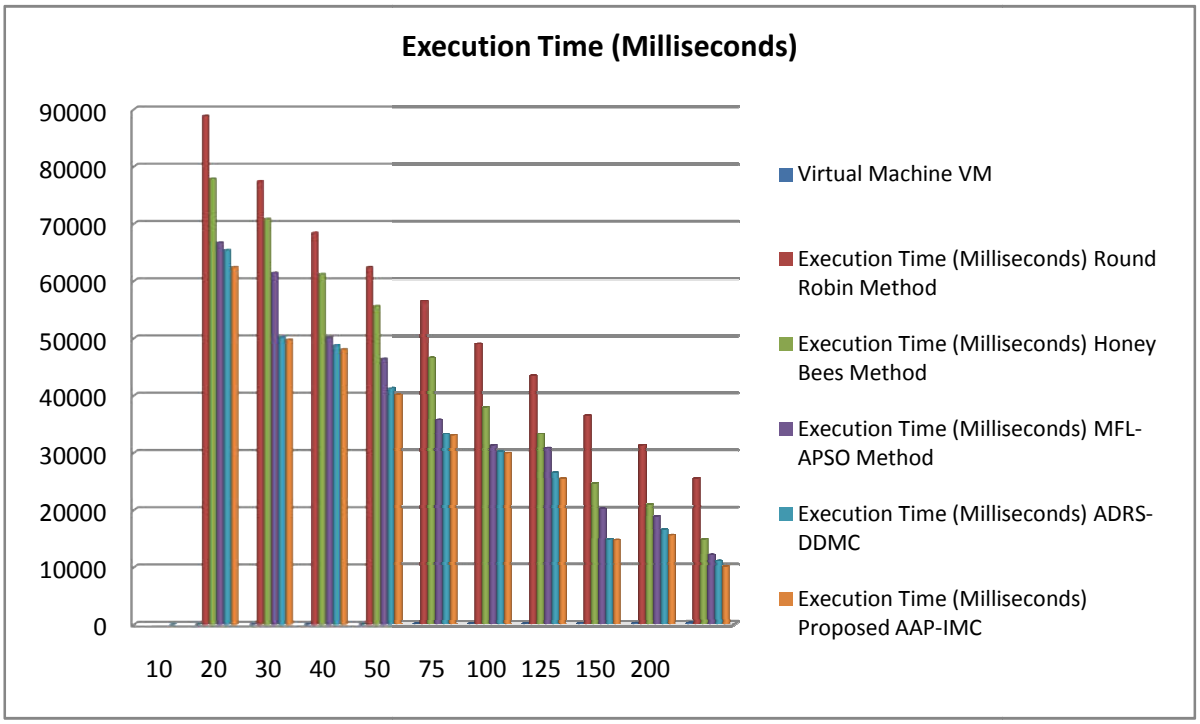

Figure 6.5.3 Average Execution Time in ms for Existing and Proposed Method

Table 6.5.3 and figure 6.5.3 shows the comparative analysis of average execution time of the 200 VMs. In figure 6.5.3 graph is plotted between the number of VMs (X-axes) and execution time in milliseconds (Y-axes). The proposed method AAP-IMC shows more efficient result than the existing method. The execution time of the VM is reduced by using AAP-IMC.

6.5.4 CPU load and CPU Utilization-In this analysis, the maximum utilization rate of the CPU and total CPU load are calculated for the proposed method.

Results for CPU Load and CPU Utilization

Work Flow Data Size in MB	CPU utilization in % For MFL-APSO				CPU utilization in % For ADRS-DDMC				CPU utilization in % For Proposed AAP-IMC			
Fuzzy Rule	Low	Medium	High	Avg %	Low	Medium	High	Avg %	Low	Medium	High	Avg %
VM	VM1	VM2	VM3		VM1	VM2	VM3		VM1	VM2	VM3	
64	11.25	24.33	36.22	23.93	16.54	18.33	19.78	18.22	16.22	17.54	18.77	17.51
128	10.25	21.33	38.9	23.49	15.44	17.44	18.77	17.22	11.23	12.1	13.22	12.18
256	9.9	21.36	37.88	23.05	8.99	24.5	30.25	21.25	11.45	21.36	28.77	20.53
512	12.35	26.45	45.66	28.15	11.42	20.41	40.15	23.99	12.35	21.64	29.78	21.26
1024	11.2	20.14	30.25	20.53	15.55	19.88	24.55	19.99	9.88	16.77	24.55	17.07
2048	9.89	20.41	25.6	18.63	8.78	19.88	24.55	17.74	6.45	16.22	18.45	13.71
3072	10.26	25.77	35.25	23.76	11.22	18.56	22.44	17.41	9.78	16.44	22.22	16.15
4096	12.3	26.44	30.21	22.98	9.66	16.44	21.55	15.88	10.22	12.44	16.55	13.07
5120	15.45	25.4	36.55	25.80	11.25	18.77	24.55	18.19	11.33	16.55	28.8	18.89
6144	14.66	24.15	32.32	23.71	12.33	24.33	31.22	22.63	11.33	22.21	30.22	21.25

Table 6.5.4 CPU Load and CPU Utilization for Existing and Proposed Method

In this analysis, the maximum utilization rate of the CPU in to evaluate the performance of the load balancing AAP-IMC algorithm. The resulting analysis gives the efficient result of the proposed approach performs under different load levels of CPU rate %. The Above results clearly show that proposed method consumes less CPU % as compared to other methods. AAP-IMC performs outstandingly in terms of CPU load and CPU utilization.

6.6 Comparative Analysis of MFL-APSO, ADRS-DDMC, Proposed AAP-IMC Algorithm-

Table 6.6 shows the overall comparative analysis of MFL-APSO, ADRS-DDMC and proposed AAP-IMC algorithm. The experimental result of table 6.6 clearly shows that the proposed algorithm AAP-IMC gives better results.

Comparative Analysis of Proposed AAP-IMC & Existing Methods

Analysis Values	Existing Methods				Proposed Method		
	HEFT	SHEFT	Round Robin	Honey Bee	MFL-APSO	ADRS-DDMC	AAP-IMC
Latency	High	High	High	Low	Low	Low	Very Low
Execution Time	High	High	High	Avg	Low	Low	Very Low
Delay	High	High	High	Avg	Low	Low	Very Low
Speed	Low	Medium	Medium	Avg	High	High	Very High
Efficiency	Less	Medium	Medium	Avg	High	High	Very High
Processing Time	1.1	1.12	1.125	1.35	1.5	1.6	1.8
Priority Based	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Scalability	Medium	High	High	High	High	High	Very High
Performance %	70%	75%	76%	80%	88%	90%	92%
Threshold Time Limit	No	No	No	No	No	No	No
Techniques Used	Heuristic Load Balancing	Advanced Heuristic Load Balancing	Round Robin	Advance Task Load Balancing	Fuzzy Logic and Workflow Based Optimization	Dynamic Distribute Data Replication Method	Combination of MFL-APSO and AAP-IMC
Waiting Time	More	More	More	Average	Less	Less	Very Less
Nature	Static	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Performance Result	Poor	Normal	Normal	Average	Better	Better	Best

Table 6.6 Comparative Analysis of MFL-APSO, ADRS-DDMC and proposed AAP-IMC

Table 6.6 shows comparative result analysis for existing MFL-APSO, ADRS-DDMC and proposed AAP-IMC. Simulation results clearly show that the proposed AAP-IMC model shows better results (better latency, execution time, delay, speed, efficiency, processing time, priority based, scalability, performance, threshold time limit, techniques used, waiting time, nature and performance) than the existing MFL-APSO, ADRS-DDMC algorithm.

CHAPTER 7

CONCLUSIONS & FUTURE WORK

This chapter highlights the main points of the thesis. It summarizes the key concepts of the chapters and elaborates on the contributions, limitations and future works of the dissertation.

7.1 CONCLUSIONS

In recent years the demand for cloud computing technology is growing rapidly. Users are demanding better performance on time and in the budget. To improve the performance of cloud computing, cloud service providers are looking for improved performance models. This research work presents a performance improvement method for cloud computing. In this work, various performance parameters for cloud computing have been discussed such as load balancing, workflows and effective resource utilization. The proposed AAP-IMC performance improvement method consists of three phases (Phase-I MFL-APSO, Phase-II ADRS-DDMC and Third final Phase AAP-IMC).

The first part of the research work presents the main research goals. The primary research theme focuses on performance improvement of cloud-based systems. In order to establish this theme, in part two of the thesis, load balancing algorithms have been depicted. The main objectives of these algorithms are to minimize the total tasks execution time. The third phase deal with the effective load balancing, optimum resource utilization of computing resources, reliability, less failure and error rates. The proposed AAP-IMC performance improvement model has been simulated in the Cloud-Sim simulator with various 200 virtual machines for validation results.

In First Phase, MFL-APSO method aims to allocate the load in VM by using various loads balancing method. To avoid heavy load in the server MFL-APSO is developed. The load balancing algorithms are architected on the mathematical apparatus of the heuristic STEM algorithm and pre-replication strategies. These algorithms could be considered as the novel ways of load balancing in cloud-based systems. Applying the STEM algorithm, the key outcomes of the experiment determine the impact of the magnitude and direction of the load on time span and performance of the system.

The main objective of the proposed MFL-APSO performance model for the cloud is to minimize the total execution time of the cloud system by considering the correct magnitude and direction of the load changes in a workflow application. In fact, the algorithm is an optimization model which minimizes the task execution time and improves the resources utilization. To implement the proposed model a Cloud-Sim and the Java-based application has been developed. An experimental result of the first phase has been compared with HEFT and SHEFT algorithm (Existing) and MFL-APSO (proposed). Model significantly reduces the total execution time of the workflow application.

The MFL-APSO (proposed method of phase-I) has three subdivisions. First subdivision deals with latency analysis. Latency time is calculated to identify the network traffic. The second subdivision deals with response time and execution time of VM. This subdivision is used to identify the speed of the cloud. The proposed MFL-APSO algorithm gives an increase in speed. The third subdivision is resource level percentage which is the most important in the proposed MFL-APSO algorithm. This subdivision deals with the percentage of load allocated to each VM by using resource level percentage metrics. This subdivision gives the efficient result. Finally, the first phase gives improved result than the existing HEFT and SHEFT algorithm. The simulation results show that proposed MFL-APSO has increased speed, decrease delay, decreases execution time and takes efficient resource level percentage for loading.

In Second Phase, the proposed ADRS-DDMC method aims to identify the reliable VMs and failure VMs. Moreover, the pre-replication analysis and results explain the importance of the anticipatory behavior of the system in terms of pre-replicating the high access probably files that can be requested in future by users. At the time of data loading in the cloud, the data is replicated in multiple VMs. It is used to give continuous service to the user at the time of disaster. If the user wants to access the stored data, only the single VM is a response to the client, not all the replicated VM. The process of the proposed algorithm is to identify one efficient VM to process the user request.

The ADRS-DDMC also identifies the failure VM. The efficient VM is identified by using reliable identification method. Phase two has three subdivisions. First subdivision deals the time

reliability; the VM which respond within the time is taken as reliable VM. The second subdivision deals the memory reliability. At any point, if two or more VMs response within the given time limit. The VM with less memory utilization is considered as reliable VM. The third subdivision deals with the previous history. If all the VMs exceed the time limit, the third subdivision is processed. In the previous history, all the existing reliable VMs are stored. The VM with the maximum count in the previous history is taken as reliable. Phase two is used to identify the reliable VM, failure VM which gives a decrease in delay and increase in speed, respectively.

The analysis has been performed to better the speed and to reduce the response time and it is compared with the parameters of LRU and LFU. We simulated our algorithm using Cloud-Sim tool. The experimental result clearly shows that proposed ADRS-DDMC algorithm performed outstandingly for cloud performance and it provides various improves results for better cloud performance such as mean job time, effective network usage and numbers of the replication needed under the defined access patterns.

In Final phase, the AAP-IMC model is the derived algorithm from MFL-APSO and ADRS-DDMC. The load balancing is done based on identifying the reliable and the failure VM. The proposed AAP-IMC (Final Phase) gives more efficient result than the MFL-APSO (Phase I) and ADRS-DDMC (Phase II) algorithms. Proposed AAP-IMC achieved better results for speed, delay, efficiency and resource optimization over MFL-APSO and ADRS-DDMC algorithms.

The final phase (AAP-IMC) has three subdivisions. First subdivision identifies the failure VM by MFL-APSO. The second subdivision assigns the priority for each VM based on the ADRS-DDMC algorithm. The VMs which have a quick response in the given time limits are taken as priority one and so on. The third subdivision assigns the load to each VM by using the AAP-IMC algorithm. To achieved better performance for an entire cloud computing system, an efficient load balancing is done by using the resource level percentage parameter. The load is balanced in all the VMs in the cloud data center. The analysis has been done to improve the cloud performance and it is compared with the performance of existing methods such as HEFT, SHEFT, Round Robin, Honey Bees, MFL-APSO and ADRS-DDMC.

The performance analysis shows that AAP-IMC increases speed and decreases execution and response time. The simulation results show that the efficient dynamic fuzzy resource level load balancing proposed model (AAP-IMC) is better than the existing algorithm. However, there is an improvement in AAP-IMC. Therefore, a better result is achieved in the proposed methodology.

The proposed algorithm achieves the system load balancing by applying self-organizing methods between overloaded VMs. This technique is structured based on communications between VMs. It helps the overloaded VMs to transfer their extra tasks to other under-loaded VMs by applying the enhanced feedbacking approach using endocrine methodology.

7.2 FUTURE WORK

The proposed new approaches in algorithms can be used in cloud IaaS. In future, the cloud environment moves to the mobile cloud, so it is necessary to include load balancing and reliability in a mobile cloud environment. In future, these algorithms can be used for mobile cloud computing and achieve more throughput and performance. We can also implement this proposed load balancing method in various real-time environments.

REFERENCES

[1]. Abbadi, I.M and Anbang Ruan, "Towards Trustworthy Resource Load Balancing in Clouds", IEEE Transactions on Information Forensics and Security Journals, Vol. 8, Issue 6 IEEE, no. 6 (2014): 973-984.

[2]. Alicherry.M. and Lakshman, T.V, "Optimizing Data Access Latencies in Cloud Systems by Intelligent Virtual Machine Placement", Proceedings IEEE Infocom, no. 4(2013): 647- 655.

[3]. Amit Kumar Das, Tamal Adhikary, Md. Abdur Razza Queue, Chong Seon Hong, "An Intelligent Approach for the Virtual Machine and QoS Provisioning in Cloud Computing", International Conference on Information Networking (ICOIN-2013), no. 1(2013), 462-467.

[4]. Anshul Rai, Ranjita Bhagwa and Saikat Guha,"Generalized Resource Allocation for the Cloud", Proceedings of the third ACM symposium on cloud computing, article no. 15, no. 10 (2012):234-240.

[5]. Anton Beloglazov and Rajkumar Buyya, "Energy-Efficient Resource Management in Virtualized Cloud Data Centers", Proceedings of ACM International Conference on Cluster, Cloud and Grid Computing, no. 6(2010): 377-384.

[6]. Apostol, E. Leordeanu, C. and Cristea, "Policy-Based Resource Allocation in Cloud Systems", Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3-PGCIC) IEEE, no. 2(2011): 285- 288.

[7]. Arunya M.R and Karthick. M, "An Adaptive Fault Tolerance Management in Cloud Computing Using Replication and Fragmentation", International Journal of Software & Hardware Research in Engineering, no. 12(2013): 79 -81.

[8]. Bakhta Meroufel and Ghalem Belalem, "Managing Data Replication and Placement based on Availability", AASRI Conference on Parallel and Distributed Computing Systems, Science Direct, no. 5(2013): 147-15

[9]. Bitar. N., Gringeri, S.Xia, T.J, "Technologies and Protocols for Data Center and Cloud Networking", IEEE communications magazines, no. 9(2013): 24-31.

[10]. Bojanova, Irena and Samba, "Analysis of Cloud Computing Delivery Architecture Models", IEEE Conference on Advanced Information Networking and Applications (WAINA), no. 3 (2011):109-118.

[11]. Brototi Mondala, Kaushik Dasgupta and Paramartha Dutta, "Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach", 2nd International Conference on Computer, Communication, Control and Information Technology (C3IT-2012) Elsevier, no. 2(2012):783-789.

[12]. Chao-Tung Yang, Wei- Li Chou, Ching-Hsien Hsu and Cuzzocrea, A.," On Improvement of Cloud Virtual Machine Availability with Virtualization Fault Tolerance Mechanism", IEEE 2nd International Conference on Net IT Trends & Cloud Computing, no. 12(2011) :122-129.

[13]. Chen, Ying, Li, Xinhua Chen and Fangning, "Overview and Analysis of Cloud Computing Research and Application", Third International Conference on Cloud Computing Technology and Science (Cloud Com) IEEE, no. 5(2011):322-329.

[14]. Chen Jung Huang, Chih-Tai Guan, Heng-Ming Chen, Yu-Wu Wang, Shun-Chih Chang, Ching-Yu Li and Chuan-Hsiang Weng, "An Adaptive Resource Management Scheme in Cloud Computing", Journal of Engineering Applications of Artificial Intelligence Science Direct, Vol. 26, no. 1(2013): 382- 389.

[15]. Chun-Wei Tsai and Joel J.P.C. Rodrigues,"Meta-heuristic Load balancing for Cloud: A Survey", IEEE Systems Journal, Vol. 8, no. 1(2014):279-297.

[16]. Daeyong Jung, Sung Ho Chin, Wang Sik Chung and Heon Chang Yu, "VM Migration for Fault Tolerance in Spot Instance Based Cloud Computing", International Conference on Grid and Pervasive Computing Springer, no. 7(2013):141- 151.

[17]. Das. P and Khilar. P.M, "VFT: A Virtualization and Fault Tolerance Approach for Cloud Computing", IEEE Conference on Information & Communication Technologies (ICT), no. 4 (2013):473-478.

[18]. Deepak Pool, Kotagiri Rama Mohana Rao and Rajkumar Buyya, "Fault-Tolerant Workflow Load Balancing Using Spot Instances on Clouds", Procedia of 14th International Conference on Computational Science Elsevier, Vol. 29, no. 5(2014): 523-533.

[19]. Dodding Probhuling L, "Load Balancing Algorithms in Cloud Computing" International Journal of Advanced Computer and Mathematical Sciences, Vol. 4, ISSN 2230-9624, no. 4 (2013): 229-233.

[20]. Dong Oh Son, Hong Jun Choi, Jae-Hyung Park and Cheol Hong Kim, "Analysis of Memory Management Policies for Heterogeneous Cloud Computing", IEEE International Conference on Information Science and Applications (ICISA), no. 6(2013):303-318.

[21]. Egwutuoha, I.P., Shaping Chen, Levy, D.; Selic, B.and Calvo, R., "A Proactive Fault Tolerance Approach to High-Performance Computing (HPC) in the Cloud", IEEE Second International Conference on Cloud and Green Computing (CGC), no. 11(2012): 268-273.

[22]. Elghoneimy, E., Bouhali, O. and Alnuweiri, H.," Resource Allocation and Load Balancing in Cloud Computing", IEEE Conference on Computing Networking and Communications (ICNC), no. 2(2013): 309-314.

[23]. Fangzhe Chang Ren, J. and Ramesh Vis Wantan," Optimal Resource Allocation in Clouds, IEEE 3rd International Conference on Cloud Computing, no. 7(2010):418-425.

[24]. Fargard, H.R., Shojaee, R., Talabani, H. and Rajabi, A., "An Analytical Model to Evaluate the Reliability of Cloud Computing Systems in the Presence of QoS Requirements", IEEE/ACIS 12th International Confer Computer and Information Science (ICIS), no. 6(2013):315-321.

[25]. Gaochao Xu, Junjie Pang and Xiaoyong, "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", IEEE Journal Tsinghua Science and Technology, Vol. 18, no. 2(2013): 34-39.

[26]. Gaurav Dhiman, Vasileios Kontorinis,"Energy-Efficient Management of Workloads in Virtualized Data Centers", Springer European Conference on Parallel Processing, no. 6(2013):557-566.

[27]. Gupta, P., Goyal, M.K. and Kumar P., "Trust and Reliability-Based Load Balancing Algorithm for Cloud IaaS", IEEE 3rd International Advance Computing Conference (IACC), no. 2(2013): 65-69.

[28]. Habib, S.M. Ries, S. and Muhlhauser M., "Towards a Trust Management System for Cloud Computing", IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (Trust Com), no, 11(2011): 933-939.

[29]. Han Xingye, Li Xinming and Liu Jinping, "Resource Management for Cloud Computing Based Information System", IEEE International Conference on Computational and Information Sciences (ICCIS), no. 12(2010):491-494.

[30]. Hongyan Cui, Yang Li, Janya Chen and Yunjie Liu, "Methods with Low Complexity for Evaluating Cloud Service Reliability", IEEE 16th International Symposium on Wireless Personal Multimedia Communications (WPMC), no. 6(2013):221-229.

[31]. Hsinyu Shin and Jenq-Shiou Leu, "Improving Resource Utilization Heterogeneous Cloud Environment", IEEE 18th Asia-Pacific Conference on Communications (APCC), no. 10 (2012): 185-189.

[32]. Hu Wu, Zhuo Tang, Renfa Li,"A Priority Constrained Load Balancing Strategy of Multiple Workflows for Cloud Computing", IEEE 14th International Conference on Cloud Computing & Advanced Communication Technology (ICACT), no. 8(2012):335-342.

[33]. Huankai Chen, Wang, F., Helian, N. and Akanmu, G., "User priority Guided Min- Min Load Balancing Algorithm for Load Balancing in Cloud Computing", IEEE Conference on Parallel Computing Technologies (PAR COMP TECH-13), no. 2(2013):111-118.

[34]. Isam Azawi Mohialdeen, "Comparative Study of Load Balancing Algorithms in Cloud Computing Environment", Journal of Computer Science & Engineering, Vol. 7, no. 1(2012): 252-263.

[35]. Jadeja, Y.Modi, K.,"Cloud Computing Concepts Architecture and Challenges", IEEE International Conference on Computing, Electronics and Electrical Technologies (ICCEET), no. 3(2012):877-880.

[36]. Janpet, J.and Yan Fu Wen,"Reliable and Available Data Replication Planning for Cloud Storage", IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), no. 3(2013):772-779.

[37]. Jenn -Wei Lin and Chien-Hung Chen and Chang, J.M, "QoS-Aware Data Replication for Data-Intensive Applications in Cloud Computing Systems", IEEE Transaction on Cloud Computing, Vol. 1, no. 6(2013):101-105.

[38]. Jhawar, R. And Piuri, V., "Fault Tolerance Management in IaaS Clouds", IEEE First AESS European Conference on Satellite Telecommunications (ESTEL), no. 10(2012): 121-131.

[39]. Jing Deng, Huang, S.C.-H., Han, Y.S. and Deng, J.H.,"Fault-Tolerant and Reliable Computation in Cloud Computing", IEEE Journal of Security & Networking, no. 12 (2010):1601-1605.

[40]. Jing Xiao, Syst and Zhiyuan Wang, "A Priority-Based Load balancing Strategy for Virtual Machine Allocations in Cloud Computing Environment", IEEE International Conference on Cloud and Service Computing, no. 11(2012) :50-55.

[41]. Jinho Hwang and Wood, T., "Adaptive Dynamic Priority Load Balancing for Virtual Desktop Infrastructures", IEEE Transactions Quality of Service (IWQoS), no. 6(2012):111-119.

[42]. Jisu Park, Heon Chang Yu and Eun Young Lee," Resource Allocation Techniques Based on Availability and Movement Reliability for Mobile Cloud Computing", Springer International Conference on Distributed Computing and Internet Technology, Vol. 7154, no. 8 (2012): 263-264.

[43]. Julia Myint and Thinn Thu Naing, "Management of Data Replication for Pc Cluster-Based Cloud Storage System", International Journal of Cloud Computing: Services and Architecture, no. 7(2011) :131-139.

[44]. Junjie Ni, Yuanqiang Huang, Zhongzhi Luan, Juncheng Zhang and Depei Qian,"Virtual Machine Mapping Policy Based on Load Balancing in Private Cloud Environment", IEEE Conference on Cloud and Service Computing (CSC), no. 12(2011):292-295.

[45]. Kashi Ventakesh Vishwanath, Nachiappan Nagappan," Characterizing cloud computing hardware reliability", Proceeding SoCC -10 the 1st ACM symposium on Cloud computing, no. 2 (2010):193-204.

[46]. Katarina Stanoevska Slava and Thomas Wozniak, "Cloud Basics-An Introduction to Cloud Computing, Journal of Grid and Cloud Computing, no. 10(2009):47-61.

[47]. Komal Mahajan, Ansuyia Makro and Deepak Dahiya,"Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud-Based Infrastructure", IEEE Conference on Cloud Computing Applications, no. 10(2011):379-394

[48]. Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong and Wang, D., "Cloud Task Load Balancing Based on Load Balancing Ant Colony Optimization", IEEE Transactions on China Grid, no. 8(2011):23-33.

[49]. Kundu, A., Banerjee, C., Guha, S.K., Mitra, A., Chakra Borty, S., Pal, C. and Roy, R., "Memory Utilization in Cloud Computing Using Transparency", IEEE Communications Journals, no. 12(2014):22-27.

[50]. Lee, R. and Bing Chiang Jeng, "Load Balancing Tactics in Cloud", IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (Cyber C), no. 10 (2011):447-454.

[51]. Liang Ma, Yueming Lu, Fangwei Zhang and Songlin Sun, " Dynamic Task Load balancing in Cloud Computing Based on Greedy Strategy, Springer International Conference on Trustworthy Computing and Services, no. 8(2013):156-162

[52]. Liang Luo, Wenjun Wu, Dichen Di, Fei Zhang, Yinzhou Yan and Yaokuan Mao, "A Resource Load Balancing Algorithm of Cloud Computing Based on Energy Efficient Optimization Methods", IEEE International Conference: Green Computing Conference (IGCC), no. 6(2012):211-216.

[53]. Limnings, N., Yuji Zhao, Yu Xiang, Tian Lan, Huang, H.H. and Subramanian, S.S, "Providing Reliability as an Elastic Service in Cloud Computing", IEEE Third Conference Cloud Applications and Challenges, no. 11(2012) :234-240.

[54]. Liyang Xie, Ningxiang Wu and Wenxue Qian,"A Perspective on Some Issues on Reliability Concept, Model and Method", 9th IEEE International Conference on Reliability, Maintainability and Safety (ICRMS), no. 6(2011):387-394.

[55]. Malik, S. and Huet, F., "Reliability-Aware Load Balancing in Cloud Computing", IEEE conference on Internet Technology and Secured Transactions, no. 12(2012):194-200.

[56]. Mayank Mishra, Anwesha Das, Purushottam Kulkarni and Anirudha Sahoo, "Dynamic Resource Management Using Virtual Machine Migrations", IEEE Journal of Cloud Computing, no. 9(2012):314-320.

[57]. Mohan N. R. and Raj, E.G., "Resource Allocation Techniques in Cloud Computing Research Challenges for Applications", Fourth IEEE International Conference on Computational Intelligence and Communication Networks (CICN), no. 11(2012):198-206.

[58]. Moreno-Volume diano, R. Montero, R.S. and Llorente, I.M., "Key Challenges in Cloud Computing: Enabling the Future Internet of Services", IEEE Internet Computing, Volume: 17, Issue: 4, no. 8(2013):121-132.

[59]. Myunghoon Jeon, Kwang Ho-Lim, Hyun Ahn and Byoung-Dai Lee, "Dynamic Data Replication Scheme in the Cloud Computing Environment", IEEE Second Symposium on Network Cloud Computing and Applications (NCCA), no. 12(2012) :40-47.

[60]. Nawsher Khan, A. Noraziah and Tutut Herawan,"Cloud Architecture with an Efficient Load Balancing Technique", Springer Journal Intelligent of Informatics, no. 8(2012):479-485.

[61]. Nuaimi, K.A. Al Ain. Mohamed, N. Nuaimi, M.A and Al Jaroodi, J., "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms", IEEE Second Symposium Network Cloud Computing and Applications (NCCA), no. 2(2012):201-226.

[62]. Olivier Beaumont, Lionel Eyraud Dubois, Hubert Larchevque, "Reliable Service Allocation in Clouds", IEEE 27th International Parallel & Distributed Processing Symposium (IPDPS) no. 10 (2013):325-333.

[63]. P.Varalakshmi, Aravindh Rama Swamy, Aswath Bala Subramanian and Palaniappan Vijay Kumar, "An Optimal Workflow Based Load Balancing and Resource Allocation in Cloud", Second International conference of Cloud Springer, no. 8 (2014):411-420.

[64]. Pannu, H.S, Jianguo Liu, Qiang Guan and Song Fu, "AFD: Adaptive Failure Detection System for Cloud Computing Infrastructures", IEEE 31st International Performance Computing and Communications Conference (IPCCC), no. 12(2012):71-80.

[65]. Paulin Florence and V. Shanthi, "Load Balancing Model Using Firefly Algorithm In Cloud Computing", Journal of Computer Science, Vol. 7, no. 2(2013):1156-1165.

[66]. Pawar C.S. and Wagh R.B., "Priority Based Dynamic Resource Allocation in Cloud Computing", IEEE International Symposium Cloud and Services Computing (ISCOS), no. 12 (2012):97-105.

[67]. Punit Gupta, Mayank Kumar Goyal, Prakash Kumar and Alok Agarwal, "Trust and Reliability-Based Load Balancing Algorithm for Cloud IaaS", IEEE 3rd International Advance Computing Conference (IACC), no. (2013):603-607.

[68]. Qiang Guan, Chi- Chen Chiu, Ziming Zhang and Song Fu, "Efficient and Accurate Anomaly Identification Using Reduced Metric Space in Utility Clouds", Published in IEEE Networking, Architecture and Storage (NAS), no. 7(2012):225-232.

[69]. Radojevic, B. and Zagar, M., "Analysis of Issues with Load Balancing Algorithms in Hosted (Cloud) Environments", 34th International Convention IEEE MIPRO, no. 5(2011):416-420.

[70]. Randles, M. Lamb, D. and Taleb Bendiab, A.," A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), no. 4(2010):551-556.

[71]. Ravi Jhawar and Marco Santana brogio, "Fault Tolerance Management in Cloud Computing: A System Level Perspective", IEEE Systems Journal, Volume: 7, Issue: 2, no. 6 (2013): 288-298.

[72]. Rimal B.P. Eunmi Choi and Lumb, I., "A Taxonomy and Survey of Cloud Computing Systems", Proceedings of IEEE Conferences INCIMS and IDC, no. 8(2009):44- 51.

[73]. Rohit O. Gupta and Tushar Champaneria, "A Survey of Proposed Job Load Balancing Algorithms in Cloud Computing Environment", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 6, no. 7(2014):782-790.

[74]. S.Sindhu and Saraswati Mukherjee, "Efficient Task Load Balancing Algorithms for Cloud Computing Environment", The Scientific World Journal Hindwai, no. 11 (2016):1-14.

[75]. Sheheryar Malik and Fabrice Huet, "Adaptive Fault Tolerance in Real-Time Cloud Computing", IEEE World Congress on Services (SERVICES), no. 7 (2011):280-287.

[76]. Shi-Chen, Jie-Wu and Zhihui-Lu, "A Cloud Computing Resource Load Balancing Policy Based on Genetic Algorithm with Multiple Fitness", IEEE communication Journal, no. 10 (2012):177-184.

[77]. Shicong Meng, Arun K. Iyengar, Isabella M, Rouvellou, Ling Liu, Kisung Lee, Balaji Palanisamy and Yuzhe Tang, "Reliable State Monitoring in Cloud Data Centers", IEEE Second International Conference on Cloud Computing (CC 12), no. 12(2012):196-205.

[78]. Souza, G.F.M., Marciano, J.P. Rodriguez C.E.P and Tomaz D.M, "Reliability Concepts Applied to Manufacturing Processes Planning", IEEE Fourth International Symposium on Uncertainty Modeling and Analysis, ISUMA 2003, no. 9(2003):92-98.

[79]. Sunil Kumar S. Manvi and Gopal Krishna Shyam, "Resource Management for Infrastructure as a Service (IaaS) in Cloud Computing: A Survey", Science Direct Journal of Networks and Computer Applications, no. 10(2013):345-350.

[80]. Tao Chen and Bassoon R., "Scalable Service Oriented Replication in the Cloud", IEEE International Conference on Cloud Computing, no. 7(2011):766-771.

[81]. Tchana, A. Brito L. and Hagimont D. "Approaches to Cloud Computing Fault Tolerance", IEEE International Conference on Computer, Information and Telecommunication Systems (CITS), no. 5(2012):111-116.

[82]. Thanadech Thanakornworakij, Raja F. Nassar, Chokchai Leangsuksun and Mihaela Paun, "A Reliability Model for Cloud Computing for High-Performance Computing Applications", Journal of European Conference on Parallel Processing Springer, no. 7(2013):474- 483.

[83]. Tharam Dillon and Chen Wu and Elizabeth Chang,"Cloud Computing: Issues and Challenges", 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), no. 4 (2010): 416-420

[84]. Tin-Yu Wu Wei-Tsong Lee, Yu- San Lin, Yih-Sin Lin, Hung-Lin Chan and Jhih Siang Huang, "Dynamic Load Balancing Mechanism Based on Cloud Storage", IEEE Conference on Computing, Communications and Applications Conference (Com-Com-Ap), no. 1(2012): 198-204.

[85]. Tiwari, A., "Dynamic Load Balancing Algorithm for Scalable Heterogeneous Web Server Cluster with Content Awareness", IEEE Conference on Trends in Information Sciences & Computing (TISC), no. 12(2010):143-148.

[86]. Tsai, C. Huang, W., Chiang, M., Chiang M. and Yang, "A Hyper-Heuristic Load Balancing Algorithm for Cloud", IEEE Transactions on Cloud Computing, no. 4(2014):173-179.

[87]. Vijindra and Sudhir Shenai,"Survey on Load Balancing Issues in Cloud Computing", Second Symposium on IEEE Network Cloud Computing and Applications (NCCA), no. 8(2012): 2881-2888.

[88]. Vilutis G., Daugirdas L., Kavaliunas R. and Sutiene K., "Model of Load Balancing and Load Balancing in Cloud Computing", IEEE 34th International Conference on Information Technology Interfaces Proceedings of the ITI 2012, no. 6(2012):117-122.

[89]. Wai-Leong Yeow, Cedric Westphal and las C. Kozat,"Designing and Embedding Reliable Virtual Infrastructures", Internal Technical Report Number (DCL-TR-2010) Docomo US Lab, no. 4(2011):57-64.

[90]. Wei Ma, Xiaoyong Li, Yong Shi and Yu Guo," TVMCM: A Trusted VM Clone Model in Cloud Computing", IEEE 6th International Conference on New Trends in Information Science and Service Science and Data Mining (ISSDM), no. 10(2012):607-611.

[91]. Weiguang Shi, M. H. MacGregor,"Load Balancing for Parallel Forwarding", IEEE IEEE/ACM Transactions on Networking, Volume: 13, Issue: 4, no. 8 (2005):790-800.

[92]. Wenhao Li, Yun Yang, Jinjun Chen and Dong Yuan, "A Cost-Effective Mechanism for Cloud Data Reliability Management Based on Proactive Replica Checking", 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CC-Grid), no. 5(2012): 217-228.

[93]. Wenhong Tian, Chengdu and Yong Zhao, "A Dynamic and Integrated Load Balancing Algorithm for Cloud Data Centers", IEEE International Conference Cloud Computing and Intelligence Systems (CCIS), no. 9(2011):311-315.

[94]. Wu, H., Tantawi, AN, Diao Y. and Wang W., "Adaptive Memory Load Management in Cloud Data Centers", IEEE IBM Journal of Research and Development, Volume: 55, Issue: 6, no. 11(2011):111-116.

[95]. Xiamen, China China, "High Reliability and Low Redundancy Storage Architecture for Cloud Computing", 2012 IEEE 7th International Conference on Networking, Architecture and Storage (NAS), no. 6(2012):109-114.

[96]. Xiao-Cheng Liu, Chen Wang, Bing Bing Zhou, Junliang Chen, Ting Yang and Albert Y. Zomaya, "Priority-Based Consolidation of Parallel Workloads in the Cloud", IEEE Transactions On Parallel And Distributed Systems, no. 9(2013): 1874-1883

[97]. Xiaonian Wu, Mengqing Deng, Runlian Zhang, Bing Zheng and Shengyuan Zhou, "Task Load balancing Algorithm based on QoS-Driven in Cloud Computing", Proceeding of Computer Science Direct Elsevier, Vol. 17, no. 8 (2013):1162-1169.

[98]. Xie Silian, "Research of Load balancing Algorithm Based on Priority in Data Nodes of Cloud Storage", IEEE Transactions on Computer Security and Cloud, no. 4 (2011):937-939.

[99]. Xie, Silian and Cheng, Yun, "A High-Reliability Replication Algorithm for Cloud Storage", IEEE Conference on Computer Science and Automation Engineering (CSAE), no. 9 (2012):506-512.

[100]. Xin Li, Zhuzhong Qian, Ruiqing Chi, Bolei Zhang and Sanglu Lu, "Balancing Resource Utilization for Continuous Virtual Machine Requests in Clouds", IEEE Communications Journals, no. 11(2012):266-273.

[101]. Xu, Gaochao, Pang, Junjie and Fu, Xiaoyong, "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", IEEE Tsinghua Science and Technology, Volume: 18, Issue: 1, no. 2 (2013):34-39.

[102]. Yaojun Han and Xue-mei Luo, "Hierarchical Load Balancing Mechanisms for Multilingual Information Resources in Cloud Computing, AASRI Proceeding Springer, Vol. 5, no. 8 (2013):268-273.

[103]. Yaqin Luo and Li Qi, "Failure-Aware Virtual Machine Configuration for Cloud Computing:" IEEE Asia-Pacific Conference: Services Computing Conference (APSCC), no. 12 (2012):125-132.

[104]. Ying Geng, Shimin Chen, Yongwei Wu, Wu, R., Gang Wen Yang and Weimin Zheng, "Locality-Aware Resource Allocation for Map Reduce in a Cloud", IEEE Parallel Processing (ICPP), no. 9(2011):101-110.

[105]. Yilei Zhang, Zibin Zheng and Michael R. Lyu and Shatin, N.T, "A Byzantine Fault Tolerance Framework for Voluntary Resource Cloud Computing", IEEE Computing Transactions, no. 6 (2011): 441-451.

[106]. Ying Shao and Xiaoming Li, "Simulation and Analysis of the Reliability of the Cloud Model-Based Automatic Environmental Monitoring System", IEEE Asia - Pacific Conference on Power and Energy Engineering Conference (APPEEC), no. 9(2011):21-27.

[107]. Younge A.J, von Las Zalewski, G, Lizhe Wang, Lopez-Alarcon S and Carithers, "Efficient Resource Management for Cloud Computing Environments", 9th Conference of IEEE on Green Computing, no. 8(2010):357-364.

[108]. Yuan-Shun Daiac, Bo Yang, Jack Dongarra, Ge Wei Zhang, "Cloud Service Reliability: Modeling and Analysis", IEEE Networking Transactions, Vol. 5, no. 11 (2010):11-17.

[109]. Yuesheng Tan, Dengliang Luo and Jingyu Wang, "CC-VIT: Virtualization Intrusion Tolerance Based on Cloud Computing", IEEE Conference on Network and Cloud Computing", no. 5(2012): 221-229.

[110]. Zhengping Wu, Nail Chu and Peng Su,"Improving Cloud Service Reliability A System Accounting Approach", IEEE Conference on Services Computing (SCC), no. 6(2012):90-97.

[111]. Zhicheng Jin, Jian Cao and Minglu Li, "A Distributed Application Component Placement Approach for Cloud Computing Environment", IEEE Conference on dependable, autonomic and Secure Computing (DASC), no. 12(2011):26-31.

[112]. Zhongyuan Lee and Ying Wang, "A Dynamic Priority Load Balancing Algorithm on Service Request Load Balancing in Cloud Computing", IEEE Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT- 2011), no. 12(2011):665-669.

[113]. Zulkar Nine, M.S.Q. Azad, M.A.K., Abdullah S. and Rahman, R.M., "Fuzzy Logic Based Dynamic Load Balancing in Virtualized Data Centers", IEEE Journal of Computing Network, no. 7(2013):117-121.

LIST OF PUBLICATIONS

Paper Published in Journals

[1]. Umesh Lilhore, Dr. Santosh Kumar, "Analysis of Performance Factors for Cloud Computing", International Journal of Information Technology and Management (IJIMTM-Ignited journal), Vol. IX, Issue No. XIV, ISSN 2249-4510, no. 9(2015): 305-310.

[2]. Umesh Lilhore, Dr. Santosh Kumar, "Survey on Load Balancing in Cloud Computing", International Journal of Information Technology and Management (IJIMTM Ignited journal), Vol. X, Issue No. XV, ISSN 2249-4510, no. 5 (2016): 224-230.

[3]. Umesh Lilhore, Dr. Santosh Kumar, "Advance Anticipatory Performance Improvement Model, for Cloud Computing", International Journal of Recent Trends in Engineering & Research (IJRTER), Volume 02, Issue 08; ISSN: 2455-1457, no. 8(2016) : 210-216.

[4]. Umesh Lilhore, Dr. Santosh Kumar, "A Novel Performance Improvement Model for Cloud Computing (IJSDR)", Volume 1, Issue 8, no. 8(2016): 410-412.

[5]. Umesh Lilhore, Dr. Santosh Kumar, "Modified Fuzzy Logic and Advanced Particle Swarm Optimization Model for Cloud Computing", International Journal of Modern Trends in Engineering and Research (IJMTER), Volume 03, Issue 08, no. 8 (2016): 230-235

[6]. Umesh Lilhore, Dr. Santosh Kumar," Anticipatory Data Replication Strategy with Dynamic Distributed Model for Cloud Computing", International Journal of Research in Applied Science & Engineering Technology (IJRASET), Volume 4 Issue VIII, no. 8 (2016): 554-559.

Paper Presented at Conferences/Seminars

[1]. Umesh Lilhore, Dr. Santosh Kumar, "Cloud Computing Performance", National Conference on New IT Trends 2015 PCCOE Engineering College Pune, no. 6 (2015): 221-227.

[2]. Umesh Lilhore, Dr. Santosh Kumar "Review of Various Load Distribution Methods for Cloud Computing, to Improve Cloud Performance", National Seminar on Security in Cyberspaces: A Challenge for Society NRI Engineering College Dept of CSE Bhopal, no. 4 (2016): 77-83.