Proactive Fault Tolerance Technique for Scientific Applications in Cloud

Thesis submitted in partial fulfillment of the requirements for the award of degree

of

Master of Engineering

in

Software Engineering

Submitted By

Suruchi

(801531014)

Under the supervision of:

Dr.Inderveer Chana

Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2017

Certificate

I hereby certify that work which is being presented in the thesis entitled," *Proactive Fault Tolerance Technique for Scientific Applications in Cloud.*", in partial fulfillment of the requirements for the award of the degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr.Inderveer Chana* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Suruchi)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

17/7/17

(Dr.Inderveer Chana)

Professor

Computer Science and Engineering Department

Acknowledgement

During the course of this thesis I have been lucky to be blessed by a lot of kind people. Let me take a moment to thank each one of them.

To begin with, I m thankful to god for his blessings and for consistently showing me the right direction.

I wish to express my sincere gratitude towards the guidance and help that I have received from my supervisor Dr.Inderveer Chana. I shall ever remain indebted for her consistent support and encouragement. She provided me help whenever needed , and also arranged for me the resources required to complete this thesis report on time.

I am thankful to Dr. Maninder Singh, Head, Computer Science and Engineering Department, Thapar University for his help and cooperation. Along with that I express my gratitude to all the staff and faculty members of Computer Science and Engineering Department, Thapar University for providing me with the facilities required for the thesis.

I would like to thank all my friends especially Resham Kaur and Pratisha Sharma for their support and suggestions. Also I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis.

I would also like to express my sincere thanks to CSIR (Council of Scientific and Industrial Research) , Government of India to carry out my research work under the project titled "Autonomic Resource Prediction and Scheduling for Cloud based Scientific Applications" with Scheme no.22(0693)/15/EMR-II.

Last but not the least I am grateful to all my family members for their inspiration and moral support which kept me motivated all the time to pursue my studies.

Suruchi

Abstract

Cloud Computing means hosted services are delivered over Internet. Instead of building their own computing infrastructures, consumers can buy the computing resources like virtual machines, storage as a utility just like an electricity. Cloud computing gives advantages to consumers like on demand service, rapid elasticity, resource sharing and pooling, scalability and many more. But there are many issues related with the cloud computing which need to be addressed properly like fault tolerance, resource scheduling and effective pricing of cloud resources etc. Making cloud environment fault tolerant is the major challenge these days. In order to improve reliability and achieve robustness in cloud computing, failures should be assessed and handled effectively.

As the cloud environment is dynamic, unpredictable system operation occurs, causing problems and failures. When Cloudlets or tasks of any application are executing on Cloud platform , then effective allocation of cloud resources like virtual machines to the cloudlets is very important such that no cloudlet gets failed or no virtual machine is left out for the execution. If there is no proper allocation of cloud resources to the cloudlets , it is considered as a big failure in the effective working of any application on Cloud platform. This fault or failure needs to be handled carefully by the fault tolerance techniques. The used techniques for the fault tolerance comes in the category of reactive (fault is handled after it has occurred) and proactive (fault is predicted before it actually happens).Much work has been done in the area of reactive techniques. The existing proactive techniques are not that much reliable in fault tolerance as compared to reactive techniques. Moreover, fault tolerance techniques work more efficiently in case of real time applications rather than scientific applications .Consequently, an approach based on proactive technique need to be proposed that can proficiently resolve the fault tolerance problem for effective allocation of cloud resources for scientific applications. This thesis focuses on providing an efficient proactive fault tolerant technique for scientific application in which above mentioned fault is predicted proactively and handled efficiently such that all cloudlets finish their tasks before or on deadline and no virtual machine has been left out for execution.

Table of Content

Content **Page No.**

List of Figures

List of Tables

Chapter 1

Introduction

Internet usage has been increased tremendously in the recent decades. Cloud Computing means delivering services on demand across the internet with the help of different models and layers of abstraction. Cloud Computing offers its services in almost every field like science, communication etc. In Cloud Computing , Internet provides remote execution of tasks located on far away distributed virtual machines. This chapter focuses on brief introduction of Cloud Computing, service models , challenges, advantages of Cloud Computing and various application areas.

1.1 Cloud Computing

Technology is growing on a rapid pace as within seven decades from the making of very first digital computer ENIAC[1],people are living in the world of internet. With the growing network bandwidth, internet is becoming an essential utility for day to day work like net banking, social networking, stock exchange etc. "Cloud" is used to represent Internet and other communications systems and also the abstraction of the underlying infrastructures involved. Cloud computing has emerged from Grid and Cluster computing. Cloud computing technology came because of the growing importance of virtualization. Service-oriented architecture is being used in the cloud and utility computing is also being adopted by the users of cloud computing on the large scale. Also the task of service providers is not just to provide services to the end users according to their demand but also to provide services in an efficient and optimized manner. Quality of Service(QoS) is always the main motive behind providing these services. To understand cloud computing, it is important to know about Grid and Cluster computing. In Grid computing, there is a collection of computer resources which are located at remote locations or servers and work together to attain single goal or task. The grid is a kind of a distributed system with non-interactive workload that involves huge number of files. The difference between Grid computing and Cluster computing is that , grid computers contain different node sets to execute different tasks/applications whereas computer cluster contains a set of loosely or tightly connected computers working together that can be seen as a single

system. Different from the Grid computers, Cluster computing involves node sets to execute the same task that is managed and scheduled by software.

Grid, Cluster and Cloud computing are interconnected as cloud computing has emerged from Grid and Cluster computing. Table 1.1 shows various characteristics of Grid, Cluster and Cloud Computing.

Table 1.1 Different characteristics of Grid, Cloud and Cluster Computing

Characteristics	**Grid Computing**	**Cloud Computing**	**Cluster Computing**
Residents	Servers and Clusters	Service User and Servers	Service user
Service Commitment	SLA's	SLA's	Limited
Resource Management	Distributed	Distributed or centralized	Centralised
Resource Allocation	Decentralised	Centralised or decentralised	Centralised
Privacy	Public or Private key for authentication	Each user with its VM's password	Old login password based
Application Areas	Scientific and complex computing applications	Dynamically and multi tenant web services.	Business oriented.

1.2 Cloud Computing Characteristics

Important characteristics of cloud computing are discussed as follows :-

i. Computing resources are always available on demand on Internet.Cloud computing gives a flexible and scalable environment [4][5].

ii. There is no capital cost involved to build cloud computing services initially. This feature is a great help for small organizations to build new services and grow their business [6][7].

iii. SaaS providers use 'Pay-as-you-go' model to provide cloud services to their consumers. This model has the principle 'the more you consume, the more you have to pay' [9][10][11].

iv. Service Level Agreement(SLA) assures quality. This agreement is signed by both consumer and service provider [8].

1.3 Cloud Computing Service Models

According to pay-as-you-go model , subscription-based services that are offered in the cloud are infrastructure ,platform, and software (applications) . Hence these services are provided according to the three different service models → Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Depending upon the type of resource customer has requested, provider responds with the associated service model. These models further comprise of different layers essential for specific service. User can access these services anywhere and at any time with the help of internet. Details of these models are as follows and detailed figure is shown as Figure 1.1. Different layers of service models are shown in Figure 1.2.

i. Infrastructure as a Service (IaaS) – This service model delivers infrastructure as a service to the customer. Also many other resources are also provided virtually like hardware, storage, network, processing power, servers etc. Companies that provide IaaS are AT&T, BlueLock, Joynet etc.

ii. Platform as a Service (PaaS)- In this service model, a platform is provided to the customer to build their applications on the web. Here at this model, users develop and deploy their applications. Also compiler, operating system, office suites, are provided

to the end user as a service. Some development models are also available as a service like WAMP and LAMP. Here 'AMP' stands for Apache servers, Mysql, PHP and 'W' stands for windows and 'L' for Linux. Platforms available in the market are Windows Azure, Google App Engine, OpenStack, Salesforce [2].

iii. Software as a Service (SaaS)- Here software is provided like any other utility. User provides application running on his client side and the user must have a client interface or a good internet connection. Here consumer does not manage any internal structure of Cloud Computing. Google, Microsoft, Amazon, Zoho etc provide SaaS [2].

Figure 1.1 Different service models in cloud computing.

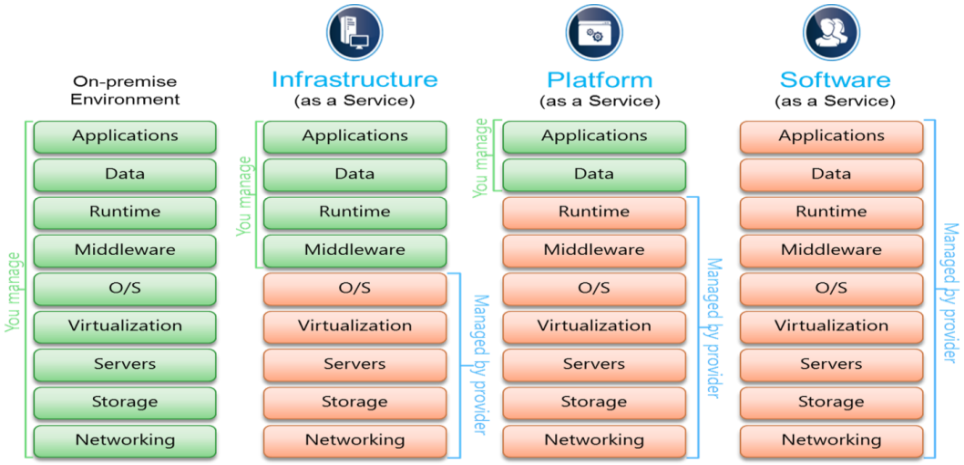


Figure 1.2 Different layers of service models in cloud computing.

1.4 Cloud Computing Deployment Models

A cloud deployment model represents a particular type of cloud environment, mainly distinguished by ownership, size, and access. Four types of cloud deployment models are-

i. Public Cloud –Here cloud resources and its services are available for the use of general public. It is managed, possesed, created by any specific organization. It is a form of providing public cloud services. Facebook is the example of public cloud. Online storage, blogs, online backups, data intensive workloads are some applications provided by public clouds [3].

ii. Private Cloud- The cloud infrastructure is used by only single organization having multiple users or business units in it. This cloud can be controlled by any organization or any third party. Companies that already own datacenter and developed IT infrastructure and have particular needs around security or performance choose to create private cloud [3].

iii. Community Cloud- The cloud infrastructure and its resources are exclusively used by a particular type of community of users sharing common concerns (e.g., goal, security policies or any compliance considerations) [3].

iv. Hybrid Cloud- It is a combination of both public and private cloud. Here end user can outsource public services and business applications can be processed by using private clouds. The main and most common attribute of hybrid cloud is elasticity and this cloud faces the problem of load balancing [3].

1.5 Cloud Computing Applications

Cloud Computing applications are increasing day by day due to its essential characteristics like flexibility and scalability. The main reasons behind success of cloud computing are-

i. Adoption of new technologies with its services.
ii. Decreasing infrastructure cost, increasing features like storage, power etc.

iii. Data size is increasing exponentially at the consumer side.

1.5.1 Scientific Applications

These applications require high computation power and throughput. Cloud serves the best platform for performing scientific applications. Some examples of scientific applications are- Montage [29,30] , Cybershake etc.

i. Montage [37] is an open source toolkit created by the NASA/IPAC Infrared Science Archive and is used for generating random mosaics of the sky using user given or software given images. Flexible Image Transport System (FITS) format is used for the images in this scientific application. At the time , when final mosaic is being generated , the shape of output image is drawn from the input images . Reprojection of all input images is done so as to obtain the same spatial scale and rotation. Background emissions that are found in the images are rectified to attain a uniform level. Finally the re-projected corrected images are all added together to generate the output mosaic. Workflow of Montage is executed in TeraGrid (Grid environment) [38].Workflow of Montage application is shown in Figure 1.3.Task dependencies between various jobs of Montage is shown in Figure 1.4.

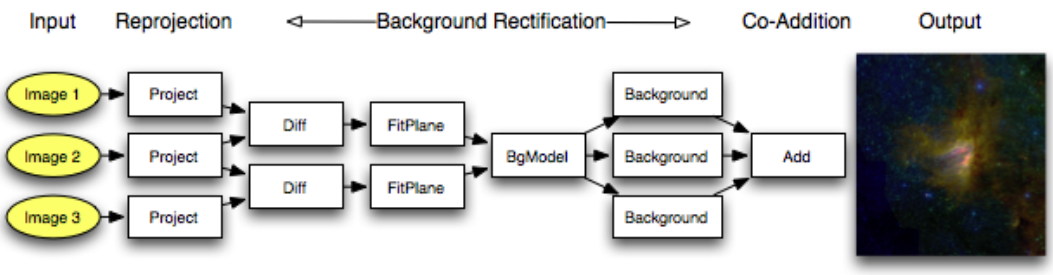


Figure 1.3 Montage Application workflow

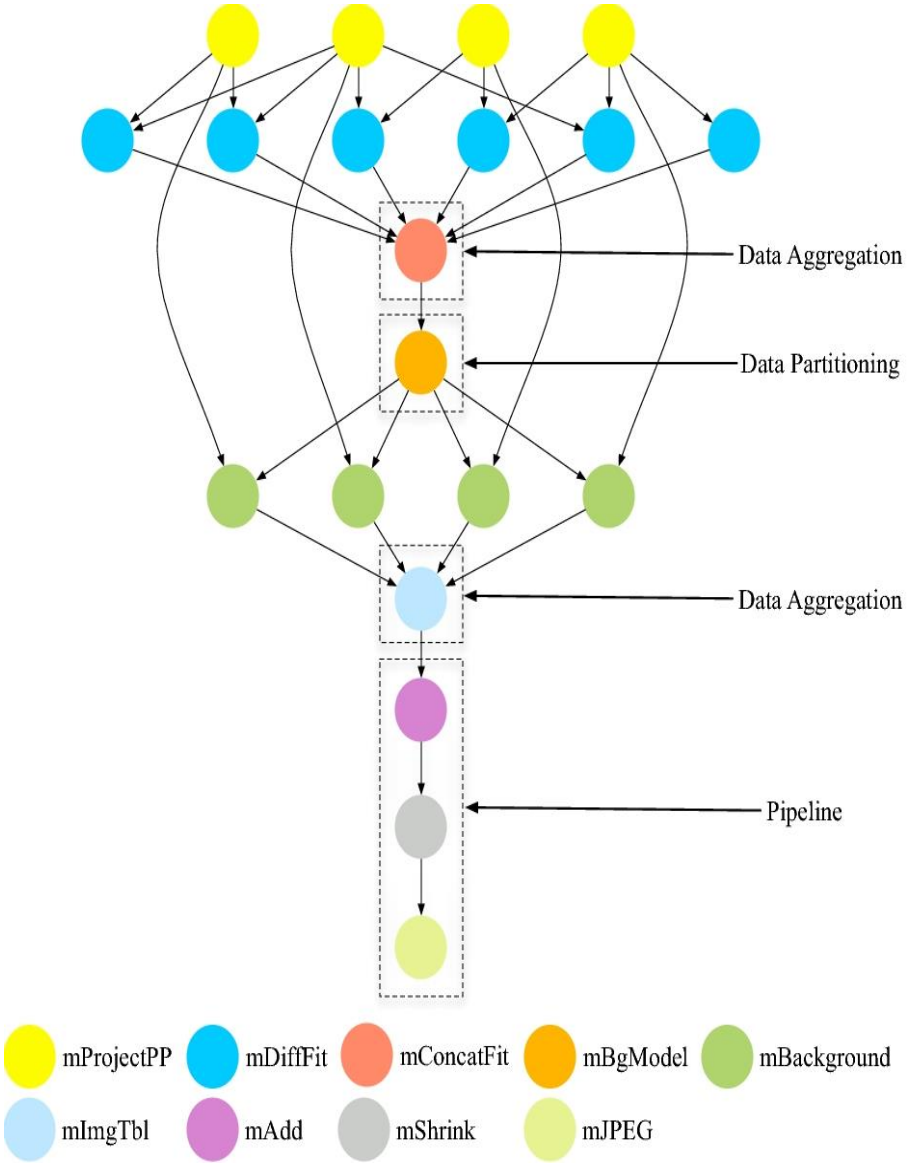


Figure 1.4 Task Dependencies among various tasks of Montage.

ii. Cybershake workflow characterises earthquake hazards by using Probalistic Seismic Hazard Analysis(PSHA) technique. Its workflows have more than 800000 jobs that execute on Pegasus Workflow Management System. Cybershake workflow is shown in Figure 1.5.

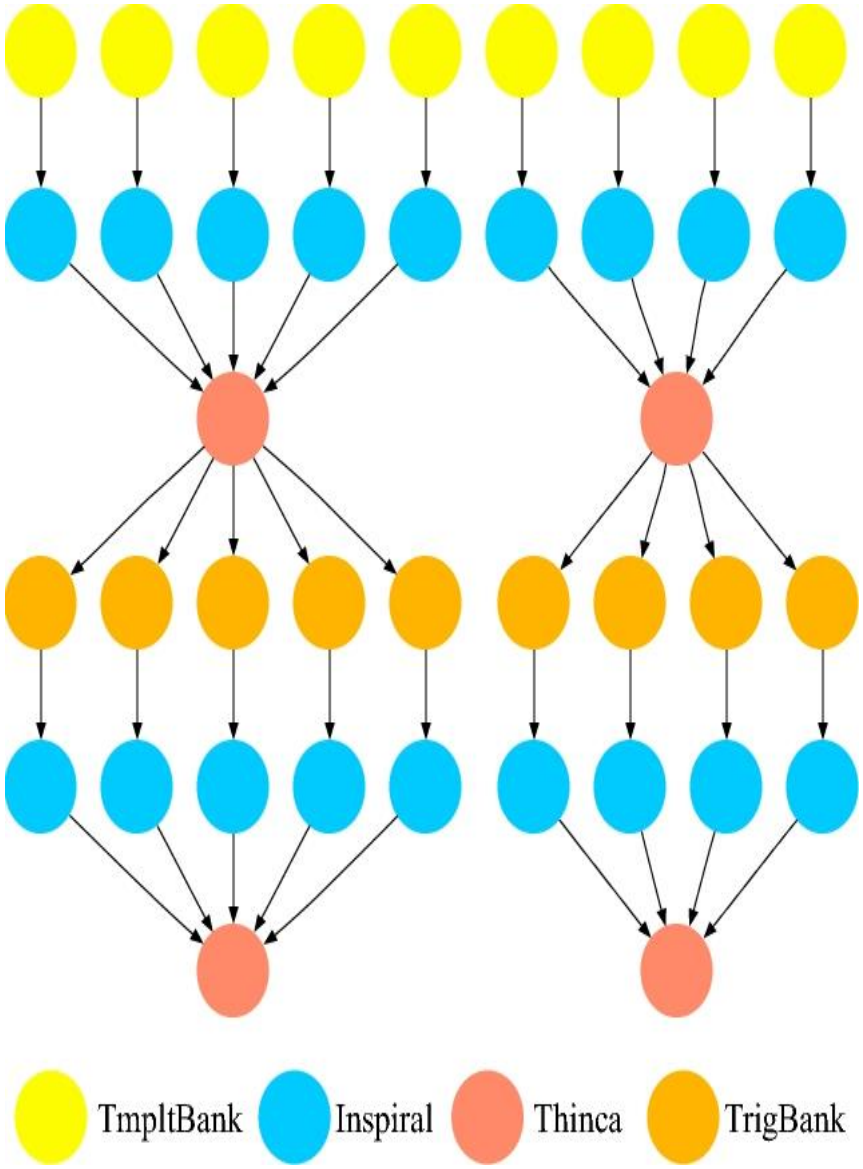


Figure 1.5 Cybershake Workflow

1.5.2 Productivity Applications

These applications use cloud computing for some commonly used desktop applications. Documents storage, office automation, website deployment like tasks can be done using cloud. Examples are Drop Box[27], RoboEarth.[28]

i. Drop Box is a service for file hosting that provides storage space, synchronization of files and many client softwares.

ii. RoboEarth is an European project to create a website for robots. It provides a giant database where sharing of messages or information can take place.

1.6 Cloud Computing Challenges

In order to support end-user applications and to give services to the customers , service providers companies like Amazon [24], HP [25], and IBM [26] have opened their cloud data centers throughout the world. Cloud has applications in almost every field from healthcare to generic text processing software. After applications get hosted on cloud platforms, anyone can have access to these services them anywhere at any time, with any device having Internet. On the back end , virtualized computers are used for their processing power, therefore speed of the application is increased for the users that pay for the services actually used. There are many challenges also in the use of these cloud services which must be handled effectively in order to make cloud services more reliable, secure, and cost-efficient. Some of the open challenges are-

i. Quality of Service (QoS) – It is necessary for the Cloud service providers (CSPs) to ensure that desirable amount of cloud resources are always provisioned in order to fulfill QoS requirements of Cloud service consumers (CSCs) like deadline, response time, and budget constraint. Basis for SLAs (Service Level Agreements) is formed by these QoS requirements and violation in any form will lead to penalty. Therefore dynamic provisioning of correct number of resources in a timely and exact manner is done to avoid or minimize the violations.

ii. Energy efficiency - Energy usage should be efficient in the cloud environment which can be done by minimizing over utilization of resources by the application. Carbon footprint should also be reduced.

iii. Security - Confidentiality (saving data from unauthorized or unauthentic access), availability (application is made unavailable to the right users by malicious users), and reliability in case of Denial of Service (DoS) attacks are very important to achieve in the cloud environment.

iv. Fault Tolerance - A system must tolerate the software or hardware faults and work properly without any failure. The main advantages of having fault tolerant systems are recovery from the failure, cost is lowered, performance metrics are improved etc.

v. Scalability - Cloud computing provides on demand scalability of its resources. But scalability can lead to vendor lock in if there is no proper interoperability. The scalability algorithms running nowadays are going to be outdated soon because cloud users are growing on a rapid pace(exponentially).Now there is a need to develop a more highly scalable cloud system.

vi. Pricing in Cloud Computing - Cloud computing provides on demand service so the users must get all the services uninterrupted for which they are paying also. Services are provided through SLA's(Service Level Agreements) which take place between customers and providers. There is a need to make cloud computing environment in which SLA violations are minimized.

vii. Load Balancing - It means load should be balanced among the cloud components. Load must be distributed evenly and uniformly among all the nodes. Load can be related to memory, network, storage or processing power. Centralised approach is not that efficient in the handling of load balancing of resources among the nodes. Therefore more decentralized approach is needed.

1.7 Research Motivation

Popularity of Cloud Computing is growing day by day and almost everyone is utilizing cloud services nowadays and this technology is giving many benefits to the users. Still there are many research challenges in the field of cloud that need to be fully solved like fault tolerance, effective scheduling of cloud resources , workflow scheduling, load balancing, security etc. The most important challenge is fault tolerance in cloud. It deals with all those techniques that are required to make a system in such a way that it can tolerate software or hardware faults that gets remained in the system after development. These techniques implement mechanisms in the system after the fault has occurred in order to prevent system from the occurrence of failure [12]. Fault tolerance provides benefits like fault recovery, reduced cost, improvement in performance metrics etc. Fault tolerant techniques are categorized as reactive and proactive. Reactive techniques work after the fault has occurred and proactive techniques predict the fault before it actually happens. Most of the work has been done in the field of reactive techniques. Also reactive techniques

are more reliable than proactive ones. Existing proactive techniques are less efficient and reliable as there is a need to decrease the fault rate in them and increase their efficiency. Therefore this research proposes a more efficient and reliable proactive fault tolerant technique.

1.8 Organisation of Thesis

Chapter 2:- This chapter includes in detail the literature survey done to study the concept of fault tolerance in the field of Cloud Computing.

Chapter 3:- This chapter provides the gap analysis , problem statement , objectives and methodology of the proposed technique.

Chapter 4:- This chapter proposes the solution to the problem in detail.

Chapter 5:- This chapter provides experimental results and comparative analysis of existing and proposed technique.

Chapter 6:- This chapter gives conclusion and future scope for the proposed technique.

Chapter 2

Literature Survey

2.1 Introduction

Cloud computing can describe on-demand administration and prerequisite of resources, and information as services on the cloud. The dynamic environment of the cloud causes various unexpected obstacles. The system's ability to respond normally to an unexpected equipment and programming fault is called fault tolerance. To realize robustness and reliability in cloud computing, it is necessary to evaluate and process obstacles effectively. Various fault detection methods and architecture models have been proposed to increase the fault tolerance capability of the cloud. This research has been paying attention on solving the problem of faults in the cloud computing. This research has been conducted to outline the work that has been done in this field which represents the basic concept of fault tolerance developed by different researchers and various algorithms used to resolve the fault tolerance problem in Cloud Computing. Reusability of IT abilities is the basic idea around which Cloud Computing is focused.

The main benefits of implementing cloud computing fault tolerance are disaster recovery, low cost, improved performance metrics, and more. Once using cloud infrastructure for real-time applications, the basic mechanism for realizing fault tolerance is a cloud node (virtual machine), transceiver sub matting node, actuator or sensor. The basic approach to accomplish fault tolerance is duplication or redundancy and this replication is performed by running software on multiple virtual machines. For replication, the price of buying cloud resources increases. Therefore , avoiding catastrophic losses is very much important.

2.2 Fault Tolerance in Cloud Computing

Fault tolerance submits to perfect and permanent operation even when there are defective components. Continuing to work satisfactorily in the presence of obstacles is the science that builds the computing system. The efficient fault tolerant system can allow fault of different types like momentary, discontinuous or stable hardware faults,

blunders in software design, operative errors, or corporal damage or externally induced upset. In a real-time application of cloud, it is performed remotely on the computing node, and the probability of error occurrence is high [44].

These error happening events enhance the desire for fault tolerance technology to achieve the efficiency of real-time computing in cloud environment. If a fault occurs in either the component (hardware) or the design method (software), an error will occur. These events enhance the need for fault tolerance technology.

2.2.1 Types of Fault Tolerance

Fault tolerance is mainly divided into two types. Fault tolerance of hardware and software [45].

2.2.1.1 Hardware fault tolerance:

Mostly computer systems nowadays get recovered by themselves from failures caused due to hardware components. In systems , each component has a back up with protected duplication or redundancy. If one component fails, the other components can perform the function. Example: Mirroring technology to repair faults in storage media. A common hardware fault tolerance approach is fault masking and dynamic recovery [46][47].

i. Fault Masking:

It is a kind of structural duplication method that absolutely identifies the errors within a block of similar components. All identical components perform the same function, their output is ranked, and errors caused by the defective module are removed. Triple module redundancy (TMR) is a very form of fault masking in which components are tripled and their output is voted. Voting process for selecting redundant components can also be tripled such that voting process can rectify the failure of personage voters.

If two components in the redundant triplet get failed and the vote is no longer valid, then TMR technique fails. Hybrid redundancy is an extension of TMR where triplexed components are always backed up with extra components and are used to replace failing components that can handle more faults. A voting system requires more than three times the hardware of a non redundant system, but it has the

advantage of continuing the calculation without interruption in the event of a failure, and you can use an existing operating system [48].

ii. Dynamic recovery:

Dynamic recovery techniques are used when only single copy of a task or any calculation is executed at one time. This technology performs self-healing method. Also it performs additional backup operations using additional spare components (proactive redundancy) along with fault masking technique. This requires a special mechanism of recovery to detect faults in the module. There is no need of switching to a redundant component in this technique because faults in modules or components are parsed and recovered by performing tasks such as rollback, initialization, retry, restart , restore operation and continue the process [48] .

2.2.1.2 Software fault tolerance:

Software failures can be handled by using static as well as dynamic approaches just like hardware fault tolerance approach. In N version programming , static redundancy is used in which a separately written program is used that executes the same operation. Each module in this technique is developed with up to N different implementations or algorithms. Every programmer does the similar task. Every version submits its output to the voter or decision-maker. The voter or the decision-maker then decides the correct output and returns it by calling it as the result of the module. Other dynamic approach is a recovery block method. In this method, a program is separated into blocks, and the acceptance test is executed after the execution of each block. In case the acceptance test fails, then redundant code block is used for the execution. In the devise diversity approach , merging of both hardware and software fault tolerance is done to make a failure free computer system using different hardware and software on redundant channels. Main goal of this technology is to withstand both hardware and software failures [49].

2.3 Fault Tolerance Techniques

Fault tolerance techniques are categorized in two types→ Reactive and Proactive. These techniques are shown in Figure 2.1.

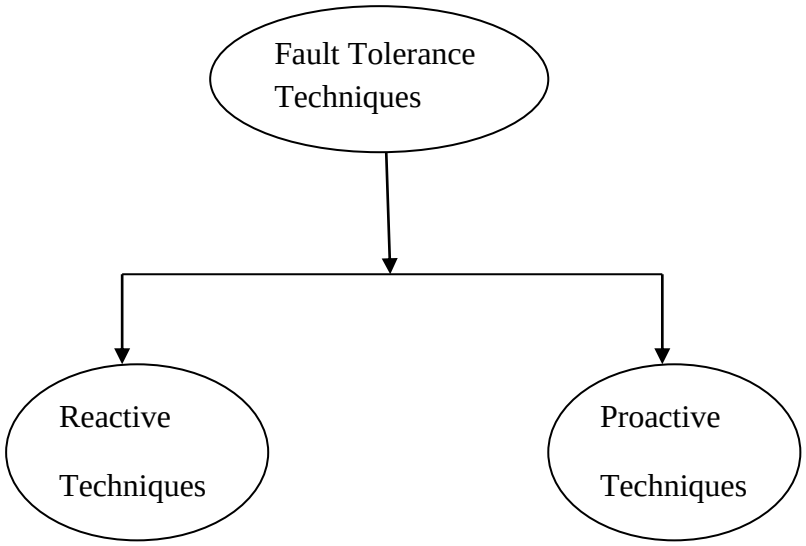


Figure 2.1 Types of Fault Tolerance Techniques

2.3.1 Proactive fault tolerance: The objective of proactive fault tolerance policy is predicting faults in advance and replace suspicious components timely by avoiding recovery from faults or errors or failures that is, they detect before actual problems occurs. This helps to prevent node failures from running parallel applications by analyzing some of the resources (tasks, processes, or virtual machines) away from the failing nodes. Several technologies working upon these policies are pre-emptive migration, software rejuvenation, and self-healing capabilities.

i. Software Rejuvenation: Here a system undergoes for periodic reboots. Rebooting the system brings the system in a new , clean and error free state [50].

ii. Migration: Preemptive migration is achieved through the feedback loop control mechanism. Applications are repeatedly identified and analyzed.

iii. Self-healing: Large tasks are divided into multiple small sub parts so as to improve the system's performance. If different instances of the application run on several different virtual machines, then automatically failure of the application instance can be handled.

2.3.2 Reactive Fault Tolerance: The goal of reactive fault tolerance policy is to reduce the effect of failures on application execution when the failure has actually occurred. It is also called on-demand fault tolerance. Various methods working on reactive policy are restart or replay , retry , checkpointing , resubmission of tasks, user defined exception handling, job migration and SGuard etc.

i. Check pointing : It is a fault tolerance technology that is implemented at task level for large-scale applications that run for a long time. In this way, after making all the changes in the system, check pointing is performed. If the task fails, rather than from the beginning, restart the system from the state of recent checkpoint [51].

ii. Job migration: Jobs may not run completely on a particular machine for some reason. When the task fails, it can be shifted to another virtual machine. HA – Proxy implements job migration.

iii. Replication: Replication refers to duplication. Different tasks are duplicated and executed on various resources to achieve successful execution and desired results. Tools such as HA-Proxy, Hadoop, AmazonEc2 replication, etc. are used for implementing replication.

2.4 Implementing Fault Tolerance using various tools

Various tools implement fault tolerance in the cloud environment. Table 2.1 compares these tools depending upon their programming language, platform used and type of application. HAProxy is used to create failure of server in the cloud [39]. SHelp is used to tolerate software failures and implementing check pointing[40] in the framework of several virtual machines. ASSURE [41] helps in introducing rescue points for tolerating failures anticipated by the developer. Hadoop [42] is used in case of data intensive applications for fault handling. For running Linux applications Amazon Elastic Compute Cloud is used and (EC2) [43] gives a virtual computing environment for fault tolerance.

Table 2.1 : Tools Used to Implement Existing Fault Tolerance Techniques

Fault Tolerance Techniques	**Policies**	**System**	**Programming Framework**	**Environment**	**Environment Fault Detected**	**Application Type**
Self Healing, Replication or Job Migration,	Reactive/ Proactive	HAProxy[31]	Java	Virtual Machine	Process/node failures	Load Balancing and Fault Tolerance
Check pointing	Reactive	SHelp[32]	SQL,JAVA	Virtual Machine	Application Failure	Fault tolerance
Check pointing, Retry, Self Healing	Reactive/ Proactive	Assure[33]	JAVA	Virtual Machine	Host, Network Failure	Fault tolerance
Job Migration,Replication,Sguard,Resc	Reactive/ Proactive	Hadoop[34]	Java,HTML,CSS	Cloud Environment	Application/ node failures	Data intensive
Replication, Sguard,Task Resubmission	Reactive/ Proactive	AmazonEC2[35]	Amazon Machine Image and Amazon Map	Cloud Environment	Node failures or application failure.	Load balancing and Fault tolerance

2.5 Existing Proposed Techniques for Fault Tolerance

Below section represents the work which has already been performed by researchers in fault tolerance.

i. Kalanirnika GR et al. [13] proposed a reactive fault tolerant approach to facilitate exploits check pointing to accept the liability. The exertion proposed a VM- μ Checkpoint framework to shield together VMs and appliances in the VMs beside ephemeral blunders. The VM-μCheckpoint apparatus was realized via CoW-PC (Copy on Write – Presave in cache) algorithm. The CoW-PC algorithm presaves every assignment consecutively on the VM's in a cache remembrance. While there were a few transitory disappointment incidents in VMs, it was remaindered and it was

improved through final presaved checkpoint from the cache reminiscence and the presaved checkpoints were obliterated robotically from the reserve recollection.

ii. Ifeanyi P. Egwutuoha et al. [14] proposed a Fault Tolerance (FT) technique for handling faults proactively in HPC systems to condense the barricade chronometer implementation instance in the occurrence of responsibilities and enlarged a standard FT algorithm. The proposed algorithm did not depend on an auxiliary node earlier to calculate or disintegrate. The authors scrutinized the dollar price of provisioning standby nodes in order to review the importance of the technique and investigational outcome acquired from an original cloud finishing location illustrated that implementation of time by the wall clock and the calculation exhaustive relevancies in cloud can be concentrated via maximum upto 30%. The occurrence of check pointing of calculation demanding applications can be reduced to 50% only with fault tolerance technique for HPC systems, measured up to existing FT techniques.

iii. Zeeshan Amin et al.[15] an assortment of fault recognition schemes and architectural models had been anticipated to enlarge fault tolerance capability of cloud. The most important rationale of this paper was to recommend an algorithm via non-natural Neural Network for burden recognition which conquer the gaps of formerly executed algorithms and afford a fault tolerant model.

iv. Dr. Chandralekha et al. [16] explored the key relationships between error tolerance and organization recital and developed metrics to measure fault tolerance within the context of system performance. By applying them to a sample mobile cloud computing environment consisting of multiple mobile nodes and demonstrate the usefulness of design metrics such as robustness. In addition, robustness was optimized using genetic algorithms and the effective fault tolerance presented by the system was evaluated.

v. Taskeen Zaidi et al. [17] deals with the thoughtful of fault tolerance procedures in cloud environment and association with diverse models on different parameters had been completed. Fault tolerance procedure was studied with the assist of well recognized object-oriented language united Modeling Language and situation diagrams were designed and legalized via the perceptions of restricted State Machine.

vi. Jhawar et al. [18] found an innovative system level modular perspective on the creation and management of cloud fault tolerance and proposed all inclusive sophisticated approach to sharing application developers and users with implementation details of fault tolerance technology using a dedicated service layer. In particular, users can themselves mention and apply the required level of fault tolerance at the service layer, and no knowledge of the assumed cloud and the fault tolerance technology available for that implementation was necessary.

vii. Kandaswamy et al. [52] introduced an efficient algorithm based on over provisioning replication as the main mechanism for fault tolerance while scheduling or mapping workflow tasks onto different resources. In over-provisioning , multiple copies of all workflow tasks (same input data-set) are executed parallelly. The aim is to maximize the success rate for the workflow task that is if one copy gets failed, other copies may be used.

viii. Nurmi et al. [53] and Schroeder et al. [54] introduced a resource reliability model for fault tolerance using Weibull distribution and taking 'mean time between failures (MTBF)' as a parameter on high performance clusters. In this model, it is reported that if the shape parameter comes less than 1, then it means that the hazard rates (the frequency with which a system or component fails) decrease with time.

ix. Reed et al.[55] identified all the reliability challenges related to large-scale HPC systems and provided with the adaptive techniques of failure detection (using performance contracts) and runtime adaptation for MPI applications.

x. Lu et al. [56] used fault injection approach to study fault sensitivity in MPI applications .

xi. Khalili et al. [57] studied the reliability of computational grids by gathering data or via deployment of monitoring infrastructure on two production grids , one of which is the TeraGrid. Inference that they have drawn was the success rates of application lies between 55% and 80%. Therefore they were motivated to include fault-tolerance mechanisms for workflows executing on computational grids.

xii. Hwang et al. [58] presented a model on failure detection service (based on notifications) and a scalable framework for tolerating grid failures using simulation as the evaluation parameter.

xiii. Alonso et al. [59] presented issues on fault-tolerance for commercial workflow management systems (WFMS) like Current commercial WFMS lack in fault-tolerance features and techniques borrowed from transactional systems like replication should be used for increased availability and better exception handling.

Table 2.2 The Existing Proposed Techniques for Fault Tolerance

Author	**Proposed technique**	**Techniques used**	**Technology employed/Strategy**	**Findings**	**Scientific Contribution**
Kalanirnika GR [13]	The CoW-pC (Copy on Write Presave in cache) algorithm	Reactive	Pre-saves all the tasks running on the VM's in a cache memory	Reduces the checkpoint overhead and execution time	Applied in Virtual Machines for against Run time errors.
Ifeanyi P. Egwutuoha [14]	Proactive Fault tolerance to HPC	Proactive	Predicts failure and takes action to lower the impact of failure.	Improves the overall execution time for computation intensive applications	Applied to computation-intensive applications running in Cloud
Zeeshan Amin [15]	Algorithm based upon Artificial Neural Network	Proactive	Estimated the expected arrival time from a virtual machine.	Evaluates the faults occurring in the system and improves the accuracy	Designed for dynamic clouds and the results are expected
Dr. Chandralekha [16]	Based on qualitative measure	Reactive	Classified the faults into three categories such as known, unknown and Undiagnosable	Excludes the parameter such as availability, reliability and fault management	The proposed work is not contributed in real world environment.
Taskeen Zaidi [17]	Proposed algorithms for global and local checkpoints	Reactive	Thread has been assigned to requested processes using simulator and allocates sub-clouds based on round robin algorithm. Checkpoints will be updated accordingly	Updates the local and global checkpoint for handling of faults	Applied test cases using the concept of Finite state Machines but do not contribute scientifically.

Ravi Jhawer [18]	Fault Tolerance Manager	Proactive	Collaborates the proposed work with delivery scheme to enable a service provider that offered fault tolerance support to client's applications	Flout Cost analysis parameter.	Delivered fault tolerance as a service
Kandaswamy [52]	Algorithm based on Over provisioning replication.	Reactive	Overprovisioning replication mechanism is used where different copies of the workflow task s(same input data-set) are executed parallelly.	Maximizes the probability of success for different workflow tasks.	Delivered fault tolerance as a service
Nurmi [53] and Schroeder [54]	Resource reliability model for fault tolerance	Reactive	Weibull distribution and taking 'mean time between failures (MTBF)' as a parameter on modern high performance clusters.	Maximizes Reliability	Delivered Reliability Model for fault tolerance.
Reed[55]	Adaptive techniques for failure detection	Reactive and Proactive	Identified reliability challenges for HPC systems and provided adaptive techniques for failure detection (using performance contracts) and runtime adaptation for MPI applications.	Reliability challenges for large-scale HPC systems	Delivered Reliability Model for fault tolerance.
Lu[56]	Fault sensitivity for MPI applications .	Reactive	Identify fault sensitivity for MPI applications	Fault sensitivity for MPI applications	Fault sensitive model
Khalili[57]	Fault-tolerance mechanisms for workflows executing on computational grids	Reactive	Studied the reliability of computational grids using data gathered via a deployment of monitoring infrastructure on two production grid	Fault-tolerance mechanisms for workflows executing on computational grids	Delivered fault tolerance as a service

			platforms,one of which is the TeraGrid		
Hwang [58]	Failure detection service (based on notifications)	Reactive	Presented a failure detection service model(based on notifications) and scalable framework for handling grid failures using simulation as the evaluation parameter.	Grid failures	Delivered fault tolerance as a service
Alonso [59]	Issues on fault-tolerance for commercial workflow management systems (WFMS)	Reactive	Presented issues on fault-tolerance for commercial workflow management systems (WFMS) like Current commercial WFMS lack in fault-tolerance features and techniques borrowed from transactional systems like replication should be used for increased availability and better exception handling.	Faults in commercial workflow management systems	Delivered fault tolerance as a service

Table 2.2 describes the earlier proposed techniques in fault tolerance.

2.6 Conclusion

This Chapter has done the study on various fault tolerant techniques and analyzed their comparison. The next chapter aims to eliminate the research gaps realized from the above study.

Chapter 3

Research Gap and Problem Statement

3.1 Research Gap

As the cloud environment is dynamic, unpredictable system operation occurs, causing problems and failures. Therefore it is essential to evaluate the faults efficiently. Fault prediction is one of the biggest challenges in securing cloud system. By the occurrence of fault in the system, performance of application executing in the cloud degrades eventually due to which it does not perform robustly [44]. Mostly used techniques for the fault tolerance comes in the category of reactive and proactive where it has been observed that reactive techniques performs effectively in comparison with proactive techniques. The existing proactive techniques are not that much reliable in fault tolerance and are not that efficient. Application failure rate is high in the existing proactive techniques. Also fault tolerant techniques work efficiently for real time applications than for scientific applications. Existing fault tolerant techniques for scientific applications are less efficient and less reliable. Therefore there is a need to make a more efficient and reliable proactive fault tolerant technique for scientific applications.

3.2 Problem Statement

There are many challenges in the area of Cloud Computing. Fault tolerance is the biggest challenge among all. Making cloud environment fault tolerant is very important. One important aspect of Cloud Computing is the effective allocation of Cloud resources like virtual machines to all the Cloudlets that are performing different tasks In the existing fault tolerant systems , there is no efficient allocation of cloud resources and failure rate for the cloudlets is very high. Also most of the fault tolerant systems handle faults reactively and efficiency rate of fault tolerance is high in case of real time applications as compared to Scientific applications . Therefore there is a need to develop a proactive fault tolerant technique for Scientific Application that can handle the fault occurred in the

effective allocation of Cloud resources proactively reducing the failure rate of cloudlets to a large extent [23].

3.3 Objectives

i. To explore various reactive and proactive fault tolerant techniques for Cloud Computing.

ii. To propose a proactive fault tolerant technique for efficient allocation of Cloud resources to all the cloudlets for a Scientific application.

iii. To implement the proposed proactive fault tolerant technique on Cloud platform.

3.4 Methodology

To accomplish the proposed technique, the below steps are followed:

i. Understand the various reactive and proactive fault tolerant techniques for real time and Scientific applications.

ii. To explore various available datasets of tasks for different Scientific applications.

iii. To setup Cloud platform for executing the cloudlets using dataset of Scientific application.

iv. Propose the proactive fault tolerant technique using Split Algorithm.

v. Compare the allocation of virtual machines and execution time of cloudlets for existing and proposed technique

Chapter 4

Proposed Proactive Fault Tolerant Technique

In cloud computing, there are number of virtual machines that execute individual tasks or cloudlets. In the previous fault tolerant systems, the allocation of virtual machines has been done on the basis of ranking. Each cloudlet has its own execution time and size. Thus, each virtual machine of the system executes cloudlet and then based upon this, an execution a table has been generated. Considering this table, the number of virtual machines who successfully completed the execution of an individual cloudlet is examined. The ranking of those machines who accomplish the execution is measured and the machine at first position will be allocated with the cloudlet such as FCFS (first successful VM in completing task). Consequently, sometimes very small sized cloudlet has been allotted to the virtual machine of very large size (executing large number of instructions per second). Moreover, there are more chances of occurrence of a condition where no suitable VM has been found for the cloudlet. Due to this problem in the existing fault tolerant systems, there is a requirement of introducing a new method where the ranking based on FCFS should not be priority.

4.1 Description of Proposed Technique

In proposed work when cloudlets arrive to the processor then the best fit approach will be applied. The best fit approach here refers to the process of allocation of suitable VM to the cloudlet which will execute it in minimum time. In case, if the best fit is not found or if any of the machine is currently not matching with the job's specifications then the proposed algorithm would eliminate this failure by processing that cloudlet in two ways, either it will split the cloudlet in two parts i.e dividing the cloudlet into two parts on the basis of its size or it will make a request for the new virtual machine. The splitting of the cloudlet is a method in which the cloudlet is divided into two parts and both parts are processed simultaneously which consumes less time and also works efficiently. In case of generating a request for the new virtual machine , memory requirement for that VM is checked. If there is no sufficient

memory for the new VM, then that cloudlet will be considered as fail, otherwise a new VM is created for that cloudlet.

4.2 Steps of the Proposed Technique

i. Enter the size of the cloudlet (taken from Scientific application -Montage) and the time needed to process that cloudlet.

ii. Execute the cloudlet on each virtual machine for obtaining the time of execution.

iii. Compare the execution time of cloudlets on different virtual machines and allot the cloudlet to the VM with minimum execution time. This VM is taken as the best fit for that particular cloudlet.

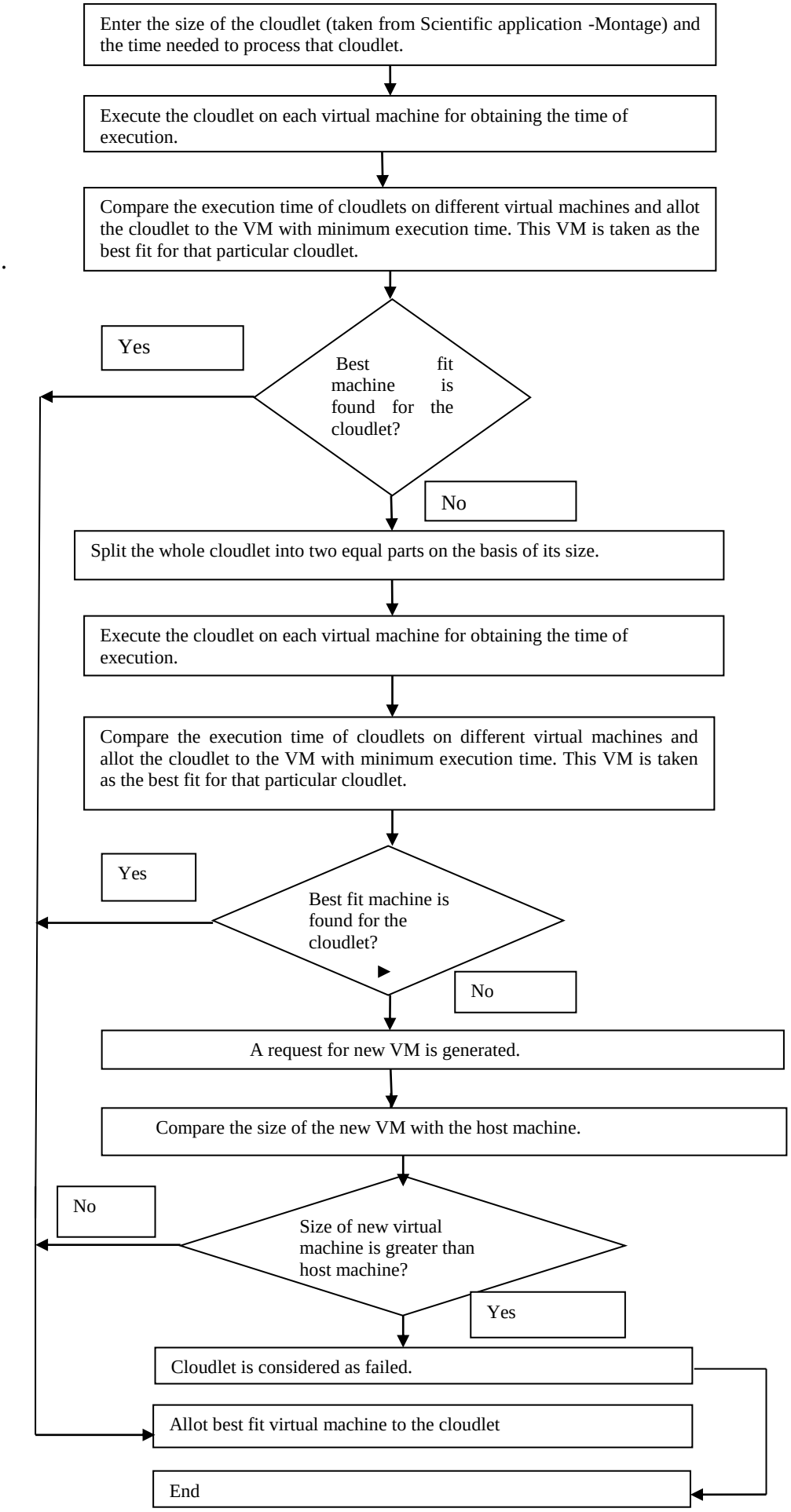
iv. If the best fit VM is not found for the execution for a particular cloudlet, then split the whole cloudlet into two equal parts on the basis of its size and then again repeat the step iii.

vi. In case after splitting of cloudlet into the new cloudlets, if any of the cloudlet does not get best fit VM, then a request for new VM is generated.

vii. Compare the size of the new VM with the host machine. If size of new VM is less than the host machine then new VM is created and allocated to the cloudlet otherwise cloudlet is considered as failed.

Flowchart for the proposed technique is shown below –

.



4.3 Scientific Application : Montage

Montage [37] is an open source toolkit created by the NASA/IPAC Infrared Science Archive and is used for generating random mosaics of the sky using user given or software given images. Flexible Image Transport System (FITS) format is used for the images in this scientific application. At the time , when final mosaic is being generated , the shape of output image is drawn from the input images . Reprojection of all input images is done so as to obtain the same spatial scale and rotation. Background emissions that are found in the images are rectified to attain a uniform level. Finally the re-projected corrected images are all added together to generate the output mosaic. The Montage application's workflow is executed in TeraGrid (Grid environment) [38]. Workflow of Montage application is shown in Figure 4.1.

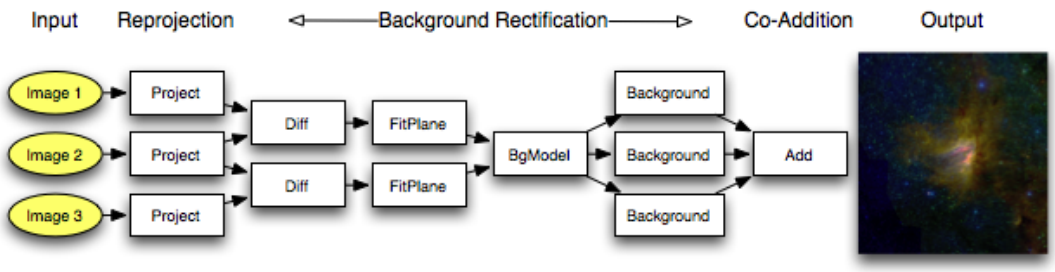
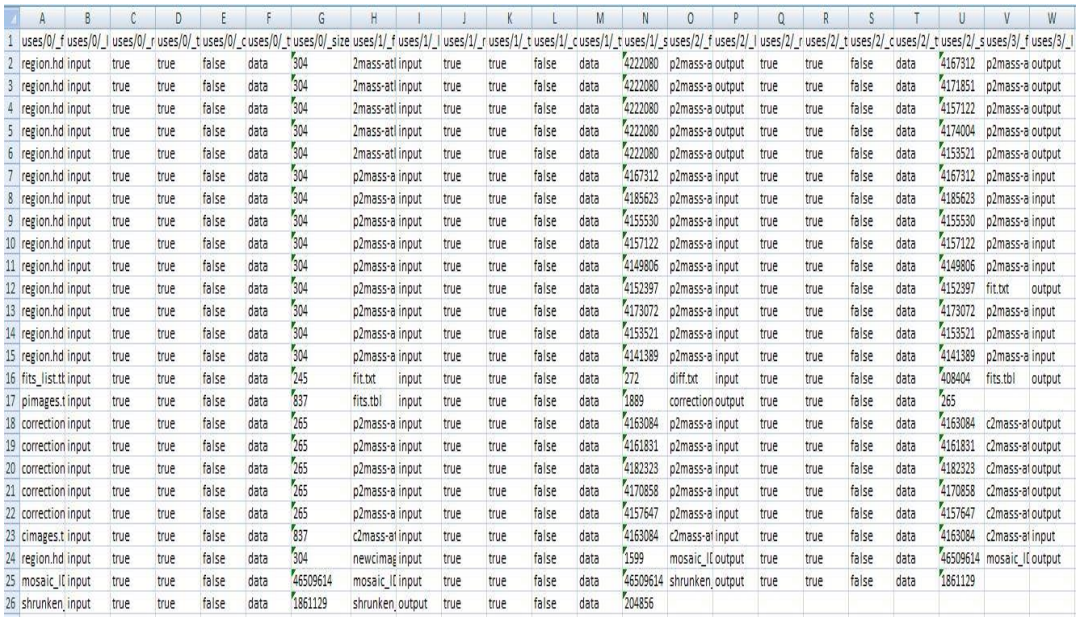


Figure 4.1 Montage Workflow

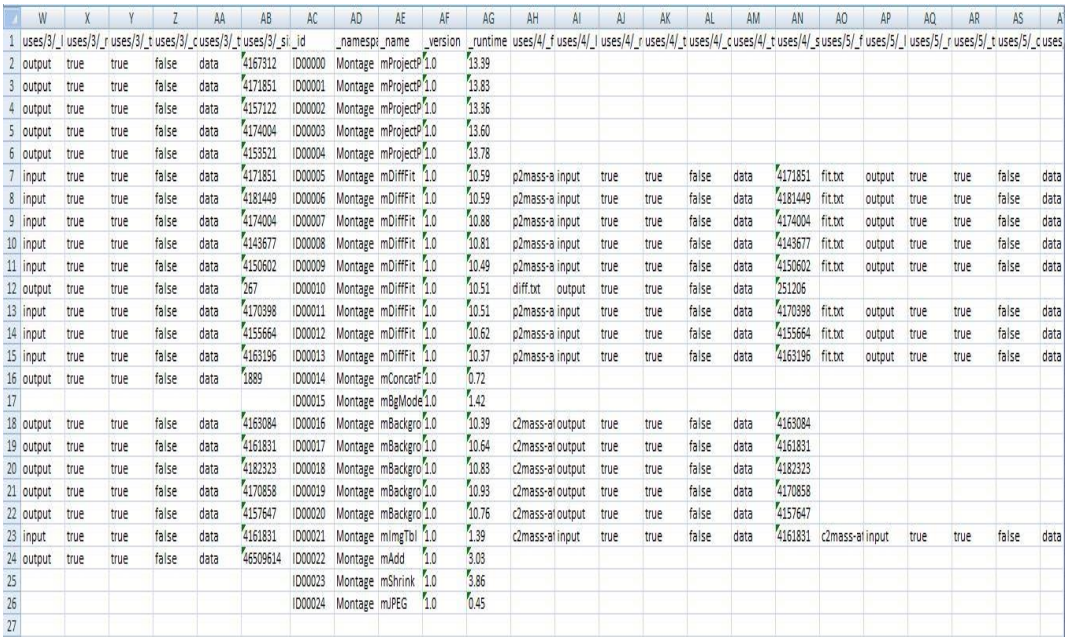
From the Montage dataset, size and execution time of the cloudlets are considered.

Screenshots of Montage Application having 25 nodes are shown as follow in Figure 4.2 and Figure 4.3.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	uses/0/_f	uses/0/_l	uses/0/_r	uses/0/_t	uses/0/_c	uses/0/_t	uses/0/_size	uses/1/_f	uses/1/_l	uses/1/_r	uses/1/_t	uses/1/_c	uses/1/_t	uses/1/_s	uses/2/_f	uses/2/_l	uses/2/_r	uses/2/_t	uses/2/_c	uses/2/_t	uses/2/_s	uses/3/_f	uses/3/_l
2	region.hd	input	true	true	false	data	304	2mass-atl	input	true	true	false	data	4222080	p2mass-a	output	true	true	false	data	4167312	p2mass-a	output
3	region.hd	input	true	true	false	data	304	2mass-atl	input	true	true	false	data	4222080	p2mass-a	output	true	true	false	data	4171851	p2mass-a	output
4	region.hd	input	true	true	false	data	304	2mass-atl	input	true	true	false	data	4222080	p2mass-a	output	true	true	false	data	4157122	p2mass-a	output
5	region.hd	input	true	true	false	data	304	2mass-atl	input	true	true	false	data	4222080	p2mass-a	output	true	true	false	data	4174004	p2mass-a	output
6	region.hd	input	true	true	false	data	304	2mass-atl	input	true	true	false	data	4222080	p2mass-a	output	true	true	false	data	4153521	p2mass-a	output
7	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4167312	p2mass-a	input	true	true	false	data	4167312	p2mass-a	input
8	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4185623	p2mass-a	input	true	true	false	data	4185623	p2mass-a	input
9	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4155530	p2mass-a	input	true	true	false	data	4155530	p2mass-a	input
10	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4157122	p2mass-a	input	true	true	false	data	4157122	p2mass-a	input
11	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4149806	p2mass-a	input	true	true	false	data	4149806	p2mass-a	input
12	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4152397	p2mass-a	input	true	true	false	data	4152397	fit.txt	output
13	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4173072	p2mass-a	input	true	true	false	data	4173072	p2mass-a	input
14	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4153521	p2mass-a	input	true	true	false	data	4153521	p2mass-a	input
15	region.hd	input	true	true	false	data	304	p2mass-a	input	true	true	false	data	4141389	p2mass-a	input	true	true	false	data	4141389	p2mass-a	input
16	fits_list.tt	input	true	true	false	data	245	fit.txt	input	true	true	false	data	272	diff.txt	input	true	true	false	data	408404	fits.tbl	output
17	pimages.t	input	true	true	false	data	837	fits.tbl	input	true	true	false	data	1889	correction	output	true	true	false	data	265		
18	correction	input	true	true	false	data	265	p2mass-a	input	true	true	false	data	4163084	p2mass-a	input	true	true	false	data	4163084	c2mass-at	output
19	correction	input	true	true	false	data	265	p2mass-a	input	true	true	false	data	4161831	p2mass-a	input	true	true	false	data	4161831	c2mass-at	output
20	correction	input	true	true	false	data	265	p2mass-a	input	true	true	false	data	4182323	p2mass-a	input	true	true	false	data	4182323	c2mass-at	output
21	correction	input	true	true	false	data	265	p2mass-a	input	true	true	false	data	4170858	p2mass-a	input	true	true	false	data	4170858	c2mass-at	output
22	correction	input	true	true	false	data	265	p2mass-a	input	true	true	false	data	4157647	p2mass-a	input	true	true	false	data	4157647	c2mass-at	output
23	cimages.t	input	true	true	false	data	837	c2mass-at	input	true	true	false	data	4163084	c2mass-at	input	true	true	false	data	4163084	c2mass-at	input
24	region.hd	input	true	true	false	data	304	newcimag	input	true	true	false	data	1599	mosaic_[0	output	true	true	false	data	46509614	mosaic_[0	output
25	mosaic_[0	input	true	true	false	data	46509614	mosaic_[0	input	true	true	false	data	46509614	shrunken_	output	true	true	false	data	1861129		
26	shrunken_	input	true	true	false	data	1861129	shrunken_	output	true	true	false	data	204856									

Figure 4.2 Dataset of Montage Application



	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	A
1	uses/3/_l	uses/3/_r	uses/3/_t	uses/3/_c	uses/3/_t	uses/3/_si	_id	_namespa	_name	_version	_runtime	uses/4/_f	uses/4/_l	uses/4/_r	uses/4/_t	uses/4/_c	uses/4/_t	uses/4/_s	uses/5/_f	uses/5/_l	uses/5/_r	uses/5/_t	uses/5/_c	uses
2	output	true	true	false	data	4167312	ID00000	Montage	mProjectP	1.0	13.39													
3	output	true	true	false	data	4171851	ID00001	Montage	mProjectP	1.0	13.83													
4	output	true	true	false	data	4157122	ID00002	Montage	mProjectP	1.0	13.36													
5	output	true	true	false	data	4174004	ID00003	Montage	mProjectP	1.0	13.60													
6	output	true	true	false	data	4153521	ID00004	Montage	mProjectP	1.0	13.78													
7	input	true	true	false	data	4171851	ID00005	Montage	mDiffFit	1.0	10.59	p2mass-a	input	true	true	false	data	4171851	fit.txt	output	true	true	false	data
8	input	true	true	false	data	4181449	ID00006	Montage	mDiffFit	1.0	10.59	p2mass-a	input	true	true	false	data	4181449	fit.txt	output	true	true	false	data
9	input	true	true	false	data	4174004	ID00007	Montage	mDiffFit	1.0	10.88	p2mass-a	input	true	true	false	data	4174004	fit.txt	output	true	true	false	data
10	input	true	true	false	data	4143677	ID00008	Montage	mDiffFit	1.0	10.81	p2mass-a	input	true	true	false	data	4143677	fit.txt	output	true	true	false	data
11	input	true	true	false	data	4150602	ID00009	Montage	mDiffFit	1.0	10.49	p2mass-a	input	true	true	false	data	4150602	fit.txt	output	true	true	false	data
12	output	true	true	false	data	267	ID00010	Montage	mDiffFit	1.0	10.51	diff.txt	output	true	true	false	data	251206						
13	input	true	true	false	data	4170398	ID00011	Montage	mDiffFit	1.0	10.51	p2mass-a	input	true	true	false	data	4170398	fit.txt	output	true	true	false	data
14	input	true	true	false	data	4155664	ID00012	Montage	mDiffFit	1.0	10.62	p2mass-a	input	true	true	false	data	4155664	fit.txt	output	true	true	false	data
15	input	true	true	false	data	4163196	ID00013	Montage	mDiffFit	1.0	10.37	p2mass-a	input	true	true	false	data	4163196	fit.txt	output	true	true	false	data
16	output	true	true	false	data	1889	ID00014	Montage	mConcatF	1.0	0.72													
17							ID00015	Montage	mBgModel	1.0	1.42													
18	output	true	true	false	data	4163084	ID00016	Montage	mBackgro	1.0	10.39	c2mass-at	output	true	true	false	data	4163084						
19	output	true	true	false	data	4161831	ID00017	Montage	mBackgro	1.0	10.64	c2mass-at	output	true	true	false	data	4161831						
20	output	true	true	false	data	4182323	ID00018	Montage	mBackgro	1.0	10.83	c2mass-at	output	true	true	false	data	4182323						
21	output	true	true	false	data	4170858	ID00019	Montage	mBackgro	1.0	10.93	c2mass-at	output	true	true	false	data	4170858						
22	output	true	true	false	data	4157647	ID00020	Montage	mBackgro	1.0	10.76	c2mass-at	output	true	true	false	data	4157647						
23	input	true	true	false	data	4161831	ID00021	Montage	mImgTbl	1.0	1.39	c2mass-at	input	true	true	false	data	4161831	c2mass-at	input	true	true	false	data
24	output	true	true	false	data	46509614	ID00022	Montage	mAdd	1.0	3.03													
25							ID00023	Montage	mShrink	1.0	3.86													
26							ID00024	Montage	mJPEG	1.0	0.45													
27																								

Figure 4.3 Dataset of Montage Application

Chapter 5

Results and Discussion

This chapter discusses about the tools needed to setup Cloud platform , implementation of the proposed technique and the corresponding results.

5.1 Tools Used to Implement Cloud Platform

5.1.1 CloudSim

CloudSim is an extensible toolkit that helps in enabling modeling of cloud resources. Simulation of cloud environment and experimentation of services in Cloud Computing can be done with the CloudSim. There is no need to focus on low level details of Cloud based infrastructure and services. It gives support to both behavior and system modelling of Cloud components like virtual machines, datacenters and resource provisioning. CloudSim features are:-

i. It provides modelling and simulation support for Cloud Computing infrastructure lying on a physical computing node.

ii. There is a hosting platform for modelling datacenteres , scheduling , service brokers and allocation policies.

iii. It provides support for modelling and simulation of energy aware computational resources.

iv. It gives modelling and simulation support for federated clouds.

v. In CloudSim, dynamic insertion of simulation elements is allowed.

vi. Support for modelling and simulation of message passing applications and data center network topologies is provided.

vii. Support for user defined policies for host allocation to virtual machines.

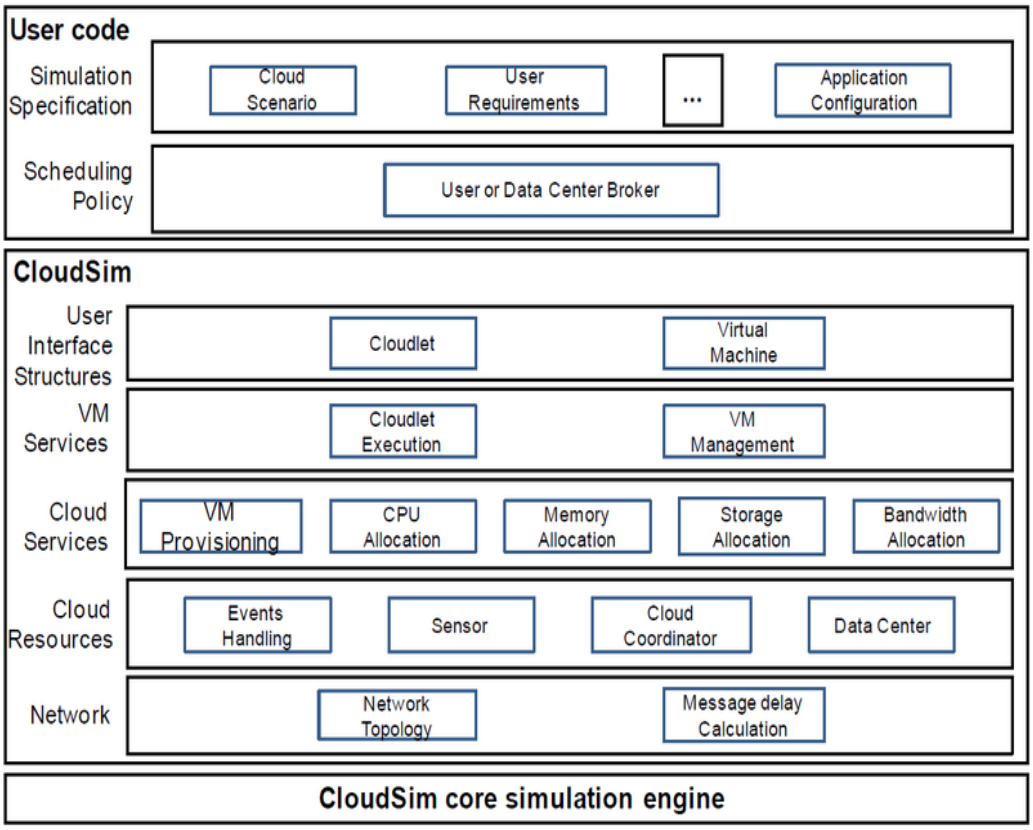


Figure 5.1 Layered CloudSim Architecture

5.1.2 NetBeans

NetBeans is a software development platform written in Java. The NetBeans platform allows applications to be developed from a set of modular software components called modules. The NetBeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++and HTML5.

5.2 Implementation of Proposed Technique

This section describes the implementation steps of the proposed technique.

i. For the first step , dataset was extracted from the WorkflowSim toolkit. From the dataset, size and execution (processing) time were considered as the parameters to implement the proposed technique.

ii. Execution of the cloudlet on each virtual machine was implemented for obtaining the time of execution as shown in Figure 5.2.

```
        }
        //print result after dividing
        for (int i = 0; i < cl.length; i++) {

            for (int i1 = 0; i1 < vm.length; i1++) {

                {

                    Cout[i1][i] = (cl[i][0] / vm[i1]);

                }

            }

        }

        //print cout array
        for (int i = 0; i < vm.length; i++) {

            for (int i1 = 0; i1 < cl.length; i1++) {
                System.out.print(Cout[i][i1] + "       ");

            }
            System.out.println("");
```

Figure 5.2 Obtaining execution time for each cloudlet for each VM

iii. Comparison of the execution time of cloudlets on different virtual machines and the allotment of the cloudlet to the VM with minimum execution time was done as shown in Figure 5.3, Figure 5.4 and Figure 5.5.

```
        System.out.println(" hell new diff array");
        for (int i = 0; i < vm.length; i++) {

            for (int i1 = 0; i1 < cl.length; i1++) {
                System.out.print(Cout[i][i1] + "       ");

            }
            System.out.println("");
        }
// set min val

        for (int i = 0; i < cl.length; i++) {

            for (int i1 = 0; i1 < vm.length; i1++) {
                if (Cout[i1][i] > 0) {
                    ef[i][0] = i1;
                    ef[i][1] = Cout[i1][i];
                    ef[i][2] = 1;
                }

            }

        }
```

Figure 5.3 Comparison of execution times of different virtual machines

```
        ///finf iffi
        for (int i = 0; i < cl.length; i++) {
            for (int i1 = 0; i1 < vm.length; i1++) {
                if (Cout[i1][i] > 0) {
                    if (Cout[i1][i] < ef[i][1]) {

                        ef[i][1] = Cout[i1][i];
                        ef[i][0] = i1;

                    }
                }
            }
        }

        System.out.println(" set efficient");
        ///// print ef
        for (int i1 = 0; i1 < cl.length; i1++) {
            System.out.println(ef[i1][0] + "-----" + ef[i1][1] + "------" + ef[i1][2]);

        }
// aloting vm to cloudlet

        for (int i1 = 0; i1 < cl.length; i1++) {
            if (ef[i1][2] == 0) {
                Valot[i1][0] = i1 + "";
                Valot[i1][1] = "Not yet ASign";
            } else {
                Valot[i1][0] = i1 + "";
                Valot[i1][1] = "" + ef[i1][0];
            }
```

Figure 5.4 Allotment of cloudlets to their best fit VM

```
        //print asign
        System.out.println(" Alotted List");
        for (int i1 = 0; i1 < cl.length; i1++) {
            System.out.println(Valot[i1][0] + "-----" + Valot[i1][1]);

        }
String fail_List="";
        // chek fail and forward them
        int f = 0;
        int m = 0;
        for (int i1 = 0; i1 < cl.length; i1++) {
            if (Valot[i1][1].equalsIgnoreCase("Not yet ASign")) {
                fail_List=fail_List+Valot[i1][0]+",";
                int x = cl[i1][0] / 2;
                nxt[f][0] = x;
                nxt[f][1] = cl[i1][1];
                nxt[f][2] = (i1 * 100) + 1;
                nxt[f + 1][0] = x;
                nxt[f + 1][1] = cl[i1][1];
                nxt[f + 1][2] = (i1 * 100) + 2;

                f = f + 2;

            } else {
                processing.Sub[m][0] = Integer.parseInt(Valot[i1][0]);
                processing.Sub[m][1] = cl[i1][0];
                processing.Sub[m][2] = Integer.parseInt(Valot[i1][1]);
                m++;

            }
```

Figure 5.5 Identification of failed cloudlets

iv. Splitting of the whole cloudlet into two equal parts on the basis of its size was done if the best fit VM is not found for the execution for a particular cloudlet as shown in Figure 5.6 ,Figure 5.7,Figure 5.8 ,Figure 5.9 and Figure 5.10.

```
        //print Operation2

JOptionPane.showMessageDialog(null, " Some Jobs are Failed to Find VM These are \n"
        + fail_List+"\n"
        + "Now apply spliting on these jobs to find appropriate Vm");


Operation2 n1 = new Operation2(nxt);

    }
}
```

Figure 5.6 Display of message for failed cloudlets

```
        }
    //print result after dividing
        for (int i = 0; i < cl.length; i++) {

          for (int i1 = 0; i1 < vm.length; i1++) {

             {

                 Cout[i1][i] = (cl[i][0] / vm[i1]) ;

              }

          }

       }
```

Figure 5.7 Splitting of cloudlets

```
0   4181499   2   83
1   4167312   0   130
2   267   4   0
3   1889   3   1
401   2080915   1   9
402   2080915   1   9
5   46509614   5   30
```

Figure 5.8 Splitting of cloudlet with id 4 creating cloudlet 401 and 402

```
        //print cout array
        for (int i = 0; i < vm.length; i++) {

            for (int i1 = 0; i1 < cl.length; i1++) {
                System.out.print(Cout[i][i1] + "        ");

            }
            System.out.println("");
        }

    //find dif
        for (int i1 = 0; i1 < cl.length; i1++) {
            for (int i = 0; i < vm.length; i++) {

               {
                    Cout[i][i1] = cl[i1][1] - Cout[i][i1] ;

                }

            }
        }

        System.out.println("  Time for every Cloudlet By each VM");
        for (int i = 0; i < vm.length; i++) {

            for (int i1 = 0; i1 < cl.length; i1++) {
                System.out.print(Cout[i][i1] + "        ");

            }
            System.out.println("");
        }
```

Figure 5.9 Calculating execution time for each cloudlet for each VM

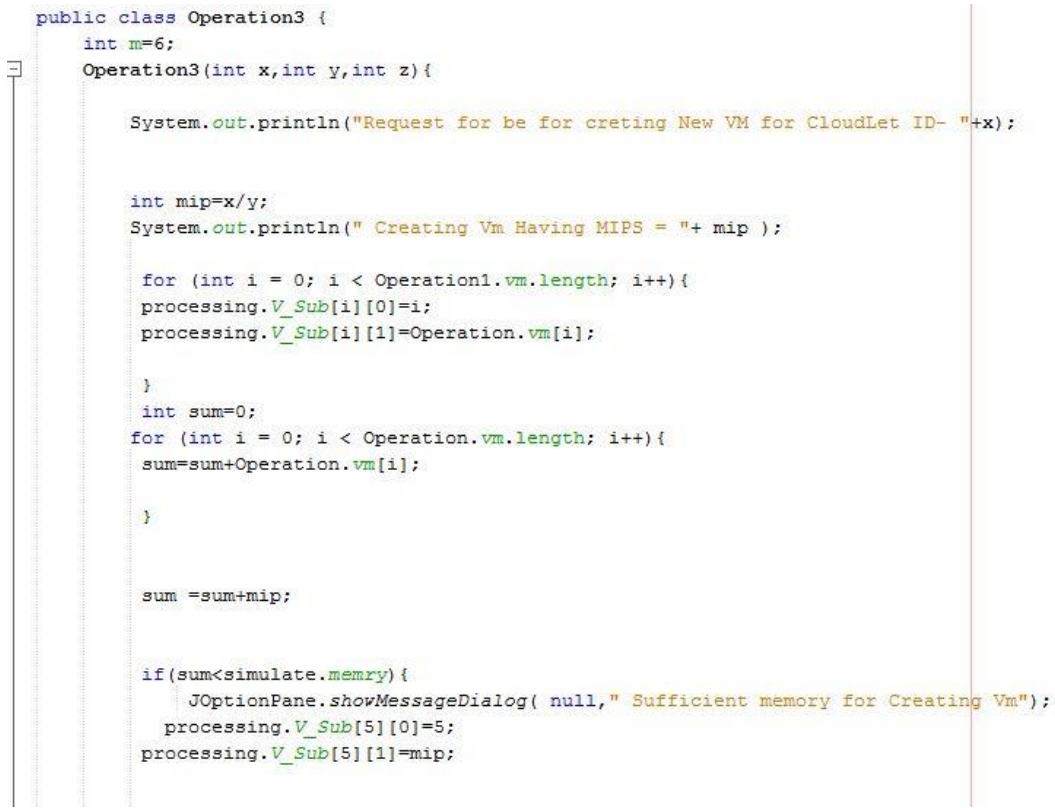
```
        System.out.println(" Print Alloted in operation 2");
       ///// print ef
          for (int i1 = 0; i1 < cl.length; i1++) {
              System.out.println(ef[i1][0]+"-----"+ef[i1][1]+"------"+ef[i1][2]);


          }
// aloting vm to cloudlet

        for (int i1 = 0; i1 < cl.length; i1++) {
        if(ef[i1][2]==0){Valot[i1][0]=i1+"";Valot[i1][1]="Not yet ASign"; }
        else{Valot[i1][0]=i1+"";Valot[i1][1]=""+ef[i1][0];}



        }

       //print asign


        for (int i1 = 0; i1 < cl.length; i1++) {
            System.out.println(Valot[i1][0]+"-----"+Valot[i1][1]);

        }

         int f=0; int m=4; int t=0; int id=0; String fail_List="1";
       for(int i1 = 0; i1 < cl.length; i1++){
       if(Valot[i1][1].equalsIgnoreCase("Not yet ASign"))

       {  //fail_List=fail_List+Valot[i1][0]+",";
           f=f+cl[i1][0];t=cl[i1][1];id=cl[i1][2 ];
```

Figure 5.10 Allotment of Cloudlets to their best fit VM.

vi. A request for new VM is generated if in case after splitting of cloudlet into the new cloudlets, if any of the cloudlet does not get best fit VM as shown in Figure 5.11 and Figure 5.12.

```
       }else{
           processing.Sub[m][0]=cl[i1][2];
            processing.Sub[m][1]=cl[i1][0];
       processing.Sub[m][2]=Integer.parseInt(Valot[i1][1]);
       m++;

       }

       }




       JOptionPane.showMessageDialog(null, " Some Jobs are Stil Failed to Find VM These are \n"
       + fail_List+"\n"
       + "Now creting new  appropriate Vm");

       Operation3 e =new Operation3(f,t,id);
       System.out.println( f+"---"+"  "+t+"-----"+ id);


   }
}
```

Figure 5.11 Identification of failed cloudlets after the splitting process



```
public class Operation3 {
    int m=6;
    Operation3(int x,int y,int z){

        System.out.println("Request for be for creting New VM for CloudLet ID- "+x);


        int mip=x/y;
        System.out.println(" Creating Vm Having MIPS = "+ mip );

         for (int i = 0; i < Operation1.vm.length; i++){
         processing.V_Sub[i][0]=i;
         processing.V_Sub[i][1]=Operation.vm[i];

         }
         int sum=0;
        for (int i = 0; i < Operation.vm.length; i++){
         sum=sum+Operation.vm[i];

         }


         sum =sum+mip;


         if(sum<simulate.memry){
             JOptionPane.showMessageDialog( null," Sufficient memory for Creating Vm");
           processing.V_Sub[5][0]=5;
         processing.V_Sub[5][1]=mip;
```

Figure 5.12 Creation of new virtual machine

vii. Comparison of the size of the new VM with the host machine was done as shown in Figure 5.13. If size of new VM is less than the host machine then new VM is created and allocated to the cloudlet otherwise cloudlet is considered as failed.

```
public class Operation3 {
    int m=6;
    Operation3(int x,int y,int z){

        System.out.println("Request for be for creting New VM for CloudLet ID- "+x);


        int mip=x/y;
        System.out.println(" Creating Vm Having MIPS = "+ mip );

         for (int i = 0; i < Operation1.vm.length; i++){
         processing.V_Sub[i][0]=i;
         processing.V_Sub[i][1]=Operation.vm[i];

         }
         int sum=0;
        for (int i = 0; i < Operation.vm.length; i++){
         sum=sum+Operation.vm[i];

         }


         sum =sum+mip;


         if(sum<simulate.memry){
             JOptionPane.showMessageDialog( null," Sufficient memory for Creating Vm");
           processing.V_Sub[5][0]=5;
        processing.V_Sub[5][1]=mip;
```

Figure 5.13 Creation of new virtual machine

5.3 Results

The proposed work provides a solution to the problem of occurrence of fault in the existing systems. Each virtual machine is allotted to the particular task in an efficient manner so that cloudlet should be executed efficiently with less time before the deadline. The experimental results acquired from the proposed technique are represented in Figure 5.14 and Figure 5.15 for the existing and proposed approach respectively.

Figure 5.14 Execution time of different cloudlets in an Existing Approach

The Figure 5.14 illustrates the execution of the traditional approach in allotting the cloudlets to the virtual machine. From the Figure, it is clear that some of the cloudlets have not been allotted to the virtual machine or no suitable virtual machine has been found for the cloudlets having id 4 and 5.These cloudlets are considered to be failed. The dialog box has presented in the figure shows the occurrence of fault in the system

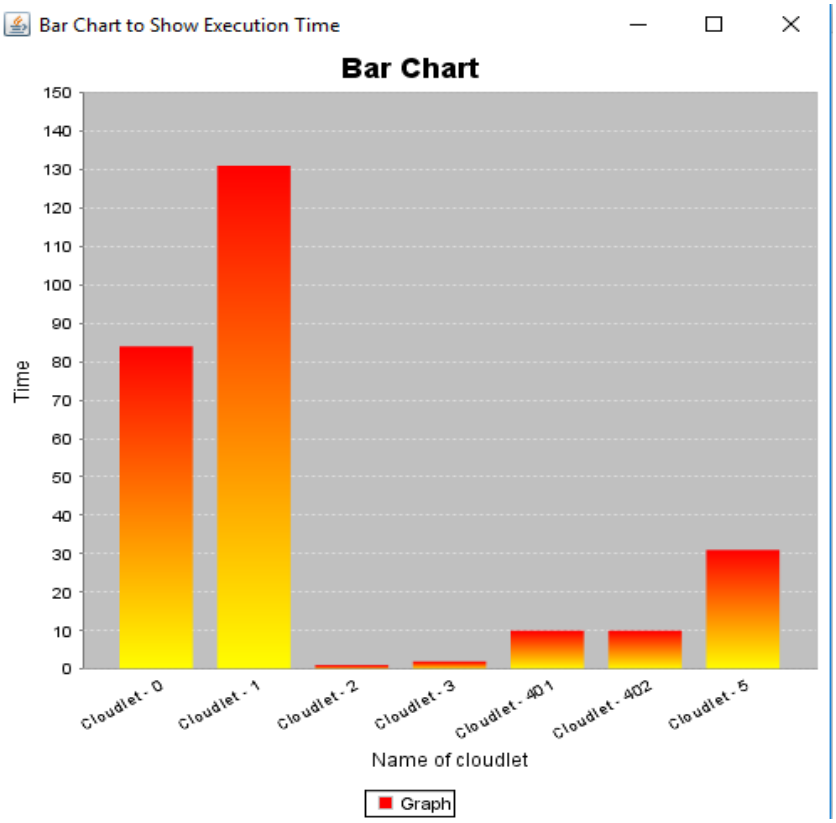


Figure 5.15 Execution time of different cloudlets in a Proposed Approach

The Figure 5.15 depicts the execution of the proposed approach where all the cloudlets are successfully initialized to each virtual machine. Thus, no fault has occurred in the system. The analysis has performed in terms of time and number of cloudlets where each VM is busy in performing its allotted task and all cloudlets have completed their tasks before or on deadline. In the proposed technique, no cloudlet has been failed thus fault has been handled proactively. Here the failure rate for the application has been decreased.

5.4 Comparitive Analysis of Failure Rates

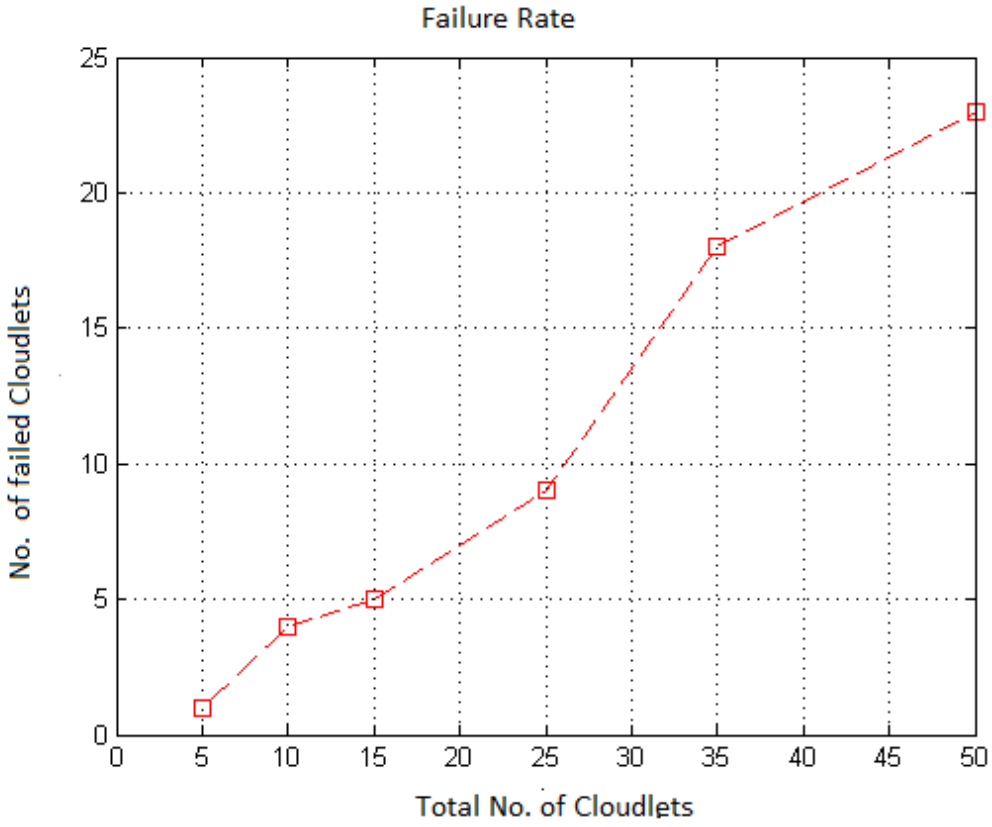


Figure 5.16 Failure Rate for Existing Technique

From the graph as shown in Figure 5.16 ,it is clear that, on increasing the Cloudlets or packets(no. of jobs) the number of failed cloudlets is also increasing or it can be said that failure rate of the application is increasing continuously. Thus existing fault tolerant technique is not reliable or efficient. Here the failure rate is growing exponentially with respect to number of total cloudlets. Therefore there is a need to develop more efficient and reliable fault tolerant technique in which rate of failure is very low or almost negligible and in which jobs can be executed successfully without

failure such that every virtual machine is busy with some task to be performed without fail.

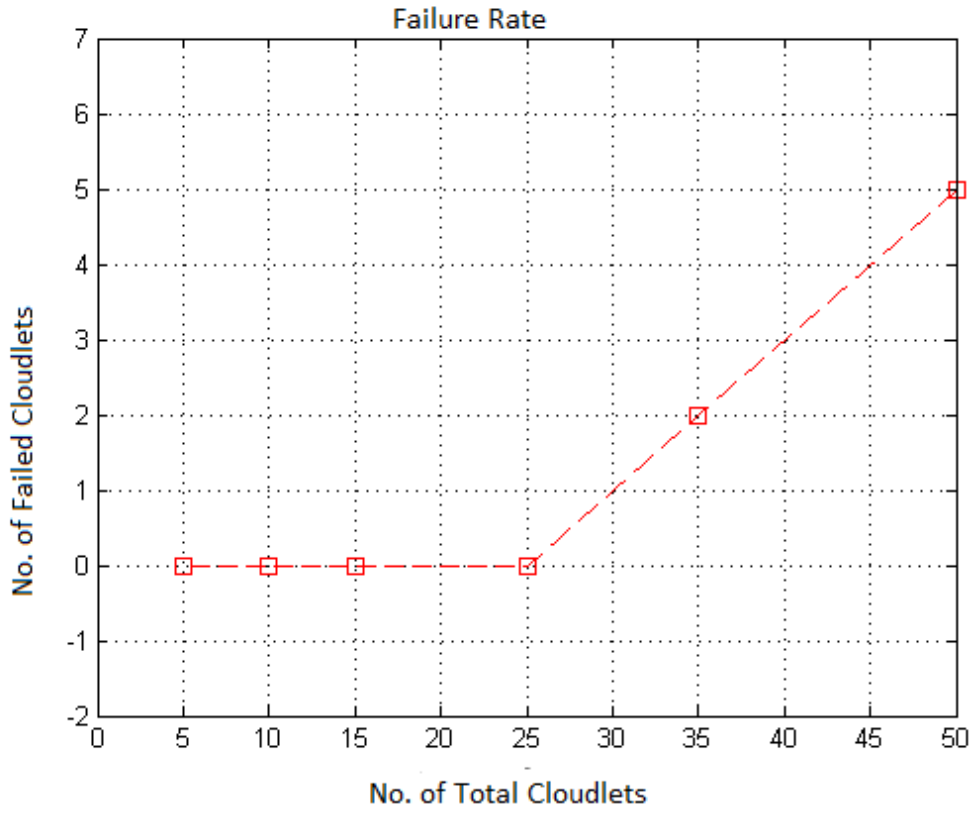


Figure 5.17 Failure Rate for Proposed Technique

From the graph as shown in Figure 5.17 , it is clear that till 25 cloudlets, number of failed cloudlets is zero. Thus no cloudlet has failed in its execution and all the tasks are performed successfully by the virtual machines before the deadline. After 25 cloudlets, failure rate begins to increase which is not that too high. Here application failure rate is growing on a small pace as compared to the existing technique in which application failure rate was growing exponentially. Thus our proposed technique is more reliable and efficient in which failure rate of cloudlets has been decreased to a large extent.

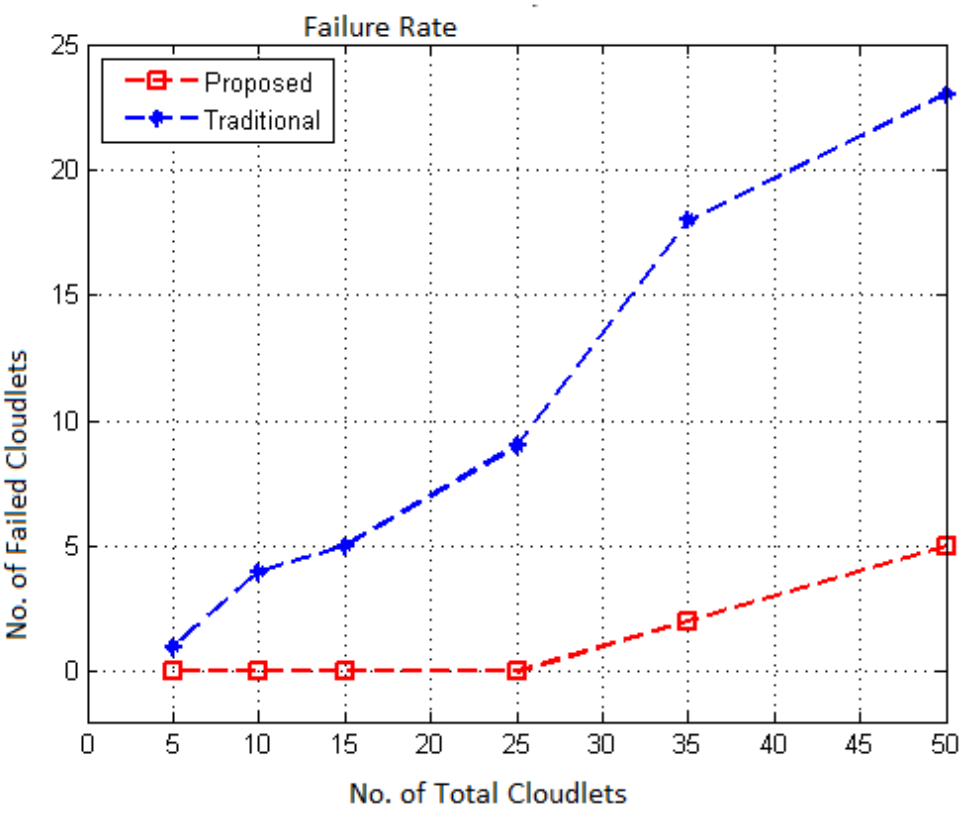


Figure 5.18 Comparison of failure rates of existing versus proposed technique

From the final graph as shown in Figure 5.18 ,it is clear that failure rate of existing fault tolerant technique is exponential with respect to the increase in number of cloudlets (jobs in an application).In the proposed technique, initially failure rate is negligible and later on increasing but on a very small rate. Thus proposed technique is more efficient and reliable as compared to existing technique.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The main goal of this research work was to propose an efficient proactive fault tolerance technique for Scientific applications in Cloud. The proposed technique satisfies all the research gaps found in the literature survey of fault tolerance techniques. The proposed technique handles the fault of allocation of virtual machines proactively such that all cloudlets complete their tasks on or before deadline and there is successful allocation of virtual machines to all the cloudlets. Probability of failure in the proposed system is less as the fault occurred in the system has been tolerated proactively. The projected work assures the reliability and the efficiency of the system where all the virtual machines are dedicated to their task and no VM has been left out.

6.2 Thesis Contribution

i. This thesis provides an efficient and reliable proactive fault tolerant technique such that there is effective allocation of virtual machines to all the cloudlets and all cloudlets finish their tasks on or before the deadline.

ii. This proactive fault tolerance technique helps in reducing the failure rate of cloudlets to a large extent.

iii. This technique ensures that all virtual machines have been successfully allotted to the cloudlets and no VM is left out.

6.3 Future Scope

i. In future the proposed work can also be enhanced with more fault tolerance capability such as by introducing the factors like cost, memory, budget etc.

ii. The proposed proactive fault tolerant technique can be made autonomic in future.

iii. Instead of simulation , the proposed technique can be used to perform real tests.

References

[1] M.H.Weik , (1961). .The ENIAC Story [Online] . Available : http ://ftp.arl.mil/mike/comphist/eniac-story.html

[2] M.Creeger,"Cloud Computing : An overview",ACM Queue,vol.7,no.5,June,2009

[3] K.Saurab,Cloud Computing,1st ed.Imdia:Wiey India,2012

[4] I.Foster,Y.Zhao, and S.Lu,"Cloud Computing and Grid Computing 360-Degree Compared," in Grid Computing Environments Workshop(GCE '08),2008.

[5] M.Armbust , A.Fox , R.Griffith , A.Joseph , R.Katz ,A.Konwinski , G.Lee ,D.Patterson ,A.Rabkin ,I.Stoica , M.Zaharia ,"Above the Clouds : A Berkeley View of Cloud Computing ," EECS Department , University of California ,Berkeley,Technical Report No.UCB/EECS-2009-28,February 2009.

[6] N.Sadashiv and D.Kumar,"Cluster,Grid and Cloud Computing:a Detailed Comparison," in the 6th International Confrence on Computer Science & Education(ICSE 2011),SuperStar Virgo,Singapore,2011,pp.477-482.

[7] S.Kumar,cloud Computing-insight into New-Era Infrastructure,1st ed.India:Wiley India,2011.

[8] R.Buyaa,C.Vecchiola, and T.Selvi,Mastering Cloud Computing,1st ed.New Delhi,India;McGraw Hill Education(India) Private Limited,2013

[9] Google App Engine Pricing [Online]Available : http://cloud.google.com/appengine/pricing

[10] Amazon Web Service Pricing[Online]. Available : http://aws.amazon.com/pricing

[11] Azure Pricing[Online]. Available : http://azure.microsoft.com/en-us/pricing

[12] Geoffroy Vallee, Kulathep Charoenpornwattana, Christian Engelmann, Anand Tikotekar, Stephen L.Scott," A Framework for Proactive Fault Tolerance".

[13] Kalanirnika G R, et al, “Fault Tolerance in Cloud Using Reactive and Proactive Techniques”, International Journal of Computer Science and Engineering Communications Vol.3, Issue 3, pp.1159-1164, 2015

[14] Ifeanyi P. Egwutuoha et al, “A Proactive Fault Tolerance Approach to High Performance Computing (HPC) in the Cloud”, IEEE, 2012.

[15] Zeeshan Amin et al, “Review on Fault Tolerance Techniques in Cloud Computing”, International Journal of Computer Applications , vol. 116 ,no. 18, April 2015.

[16] Dr. Chandralekha et al, “Effective Fault Tolerance Management in Cloud” International Journal of Emerging Technology and Advanced Engineering , vol. 6, Issue 3, March 2016.

[17] Taskeen Zaidi et al, “Modeling for Fault Tolerance in Cloud Computing Environment”, Journal of Computer Sciences and Applications, vol. 4, no. 1,pp 9-13, 2016

[18] R. Jhawar, et al, “Fault tolerance management in cloud computing: A system-level perspective. Systems Journal”, IEEE, 7(2), pp 288-297, 2013.

[19] Deelman, E., Singh, G., Su, M., et al. 2005. Pegasus: a framework for mapping com¬plex scientific workflows onto distributed systems. Scientific Computing 13:219–237.

[20] Oinn, T., Greenwood, M., Addis, M., et al. 2006. Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18:1067–1100.

[21] Taylor, I., Shields, M., Wang, I., et al. 2007. The Triana Workflow Environment: Architecture and Applications. In Workflows for E-Science, ed. I. J. Taylor, E. Deelman, D. B. Gannon, et al., 320–339. London: Springer.

[22] Pandey, S., Karunamoorthy, D., and Buyya, R. 2011. Workflow engine for clouds. In Cloud Computing: Principles and Paradigms, ed. R. Buyya, J. Broberg, and A. Goscinski, 321–344. New York: Wiley.

[23] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou, Weizhong Qiang, Gang Hu, "SHelp: Automatic Selfhealing for Multiple Application Instances in a Virtual Machine Environment", IEEE International Conference on Cluster Computing, 2010.

[24] J. Varia, Best practices in architecting cloud applications in the AWS Cloud. In: R. Buyya, J. Broberg, A. Goscinski (eds.), Cloud Computing: Principles and Paradigms, Wiley Press, 2011.

[25] HP. Utility Computing. http://www.hp.com/go/utility/.

[26] IBM Smart Cloud. http://www.ibm.com/cloud-computing/.

[27] Dropbox inc.Dropbox.[Online] available ; http://www.dropbox.com/

[28] E.Guizzo,"Robots with their heads in the cloud,"IEEE Spectrum,2011.

[29] The Montage project, http://montage.ipac.caltech.edu/.

[30] G. B. Berriman, J. C. Good, D. Curkendall, J. Jacob, D. S. Katz, T. A. Prince, andR. Williams. Montage: An on-demand image mosaic service for the NVO,Astronomical Data Analysis Software and Systems (ADASS) XII, 2002.

[31] Taskeen Zaidi et al, "Modeling for Fault Tolerance in Cloud Computing Environment", Journal of Computer Sciences and Applications, vol. 4, no. 1,pp 9-13, 2016

[32] http://haproxy.1wt.eu/download/1.3/doc/configuration.txt

[33] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou, Weizhong Qiang, Gang Hu, "SHelp: Automatic Selfhealing for Multiple Application Instances in a Virtual Machine Environment", IEEE International Conference on Cluster Computing, 2010.

[34] S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, and A. D. Keromytis, "ASSURE: Automatic Software Self-healing Using REscue points", Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS'09), ACM Press, March 7-11, 2009, Washington, DC, USA, pp.37-48.

[35]HadoopMapReduceTutorial.http://hadoop.apache.org/core/docs/current/mapredtutorial.html.

[36] AmazonElasticComputeCloud(EC2) http://www.amazon.com/ec2/

[37] G.B. Berriman, et al. Montage: a Grid enabled engine for delivering custom science-grade mosaics on demand, in: SPIE Conference 5487: Astronomical Telescopes, 2004.

[38] Maq: mapping and assembly with qualities. http://www.maq.sourceforge.net.

[39] Yang Zhang1, Anirban Mandal2, Charles Koelbel1 and Keith Cooper," Combined Fault Tolerance and Scheduling Techniques for Workflow Applications on Computational Grids "in 9th IEEE/ACM international symposium on clustering and grid, 2010.

[40] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zho Weizhong Qiang, Gang Hu, "SHelp: Automatic Selfhealing for Multiple Application Instances in aVirtual Machine Environment", IEEE International Conference on Cluster Computing, 2010.

[41] S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, and A. D. Keromytis, "ASSURE: Automatic Software Self-healing Using REscue points", Proceedings of the 14thInternational Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'09), ACM Press, March 7-11,2009,Washington, DC, USA, pp.37-48 .

[42] HadoopMapReduceTutorial.http://hadoop.apache.org/co. re/docs/current/mapredtutorial.html.

[43] AmazonElasticComputeCloud(EC2) http://www.amazon.com/ec2/

[44] Patra, P. K., Singh, H., & Singh, G. (2013). Fault Tolerance Techniques and Comparative Implementation in Cloud Computing. International Journal of Computer Applications, 64(14).

[45] Bala, A., & Chana, I. (2012). Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing. International Journal of Computer Science Issues (IJCSI), 9(1).

[46] http://en.wikipedia.org/wiki/Fault-tolerant_computer_system

[47] http://www.cs.ucla.edu/~rennels/article98.pdf

[48] Qin Zheng, "Improving MapReduce Fault Tolerance in the Cloud" ©2010 IEEE.

[49] Amritpal Singh, Supriya Kinger, "An Efficient Fault Tolerance Mechanism Based on Moving Averages Algorithm" © 2013,.

[50] M.Armbrust, A.Fox, R. Griffit,et al., "A view of cloud computing", Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[51] Golam Moktader Nayeem, Mohammad Jahangir Alam," Analysis of Different Software Fault Tolerance Techniques", 2006.

[52] Gopi Kandaswamy, Anirban Mandal, and Daniel A. Reed. Fault tolerance and recovery of scientific workflows on computational grids. In CCGRID '08: Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID), pages 777–782, Washington, DC, USA, 2008. IEEE Computer Society.

[53] Daniel Nurmi, John Brevik, and Rich Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In In Euro-Par05, pages 432–441. Springer, 2005.

[54] Bianca Schroeder and Garth A. Gibson. A large-scale study of failures in high-performance computing systems. In DSN '06: Proceedings of the International Conference on Dependable Systems and Networks, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.

[55] D. A. Reed, C. Lub, and C. L. Mendes, "Reliability challenges in large systems," Future Generation Computer Systems, vol. 22, no. 3, pp. 293-302, Feb. 2006.

[56] C. Lu, and D.A. Reed, "Assessing Fault Sensitivity in MPI Applications," Proceedings of the 2004 ACM/IEEE conference on Supercomputing, Nov. 2004.

[57] O. Khalili, J. H. Olschanowsky, C. Snavely and H. Casanova, “Measuring the Performance and Reliability of Production Computational Grids,” Proceedings of the 7th IEEE/ACM International Conference on Grid Computing, Sep. 2006.

[58] S. Hwang, and C. Kesselman, “A flexible framework for fault tolerance in the grid,” Journal of Grid Computing, vol. 1, no. 3, pp. 251-272, 2003.

[59] G. Alonso, C. Hagen, D. Agrawal, A. E. Abbadi, and C. Mohan, “Enhancing the fault tolerance of workflow management systems,” IEEE Concurrency, vol. 8, no. 3, pp. 74-81, 2000.

List of Publications

[1] Suruchi Talwani and Inderveer Chana ," Fault Tolerance for scientific applications in cloud", in *2nd International Conference on Telecommunication and Networks (TEL-NET 2017)* , Amity University Uttar Pradesh, Noida, India , to be held in August 2017. [Accepted].