Provisioning Hybrid Fault Tolerance Using Reactive and Proactive Approaches in Cloud Computing

Falema Garedow Herpa

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of The Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February 2023

Adama, Ethiopia

Provisioning Hybrid Fault Tolerance Using Reactive and Proactive Approaches in Cloud Computing

Falema Garedow Herpa

Advisor: Dr-Ing Frezewd Lemma

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of The Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February 2023

Adama, Ethiopia

Declaration

I declare that this MSc Thesis entitled "**Provisioning Hybrid Fault Tolerance Using Reactive and Proactive Approaches in Cloud Computing"** is my own work and has not been submitted to any university for similar purpose. The references used in this Research are duly recognized by proper citations.

Falema Garedow Herpa _____ _____

Name of student Signature Date

Approval Page for thesis

I/we hereby certify that the recommendation and suggestion given by the research review committee are appropriately incorporated into the final thesis entitled "**Provisioning Hybrid Fault Tolerance Using Reactive and Proactive Approaches in Cloud Computing**" by Falema Garedow Herpa.

Dr-Ing Frezewd Lemma _____ _____

Major Advisor/Supervisor Signature Date

Table of contents

List of Tables

List of Figures

List of Equation

Abstract

This particular research work is intend to provisioning a hybrid fault tolerance by combining the proactive and reactive fault tolerance approaches to address the problem that arises in the cloud environment that leads to system failures and service outages. Therefore, the objective of this work is to analyze the existing fault tolerance techniques and models in cloud computing and integrate the two fault tolerance approaches to ensure that the cloud environment is reliable and available by developing an appropriate algorithm that can solve this problem. Therefore, this study focuses on provisioning hybrid fault tolerance to ensure the quality of service in the cloud computing environment by realizing the reliability, of the cloud system by improving execution time, throughput, and migration time. Therefore, this study contributes a Threshold-Based Adaptive Hybrid Fault Tolerance (TBAHFT) approach in the cloud by amalgamating proactive and reactive approaches. Furthermore, in this study, VM migration and replication are being utilized in the proactive approach, whereas retry is used as a reactive approach. Additionally, experimental validation is performed by comparing three benchmark fault tolerance approaches by incorporating the first fit and least full host selection policies. The obtained experimental results of the proposed approach are presented in terms of different metrics such as failure rate, throughput, migration, and execution time using CloudSim simulator.

Keywords: Throughput, failure rate, execution time, migration time, availability, reliability, hybrid fault tolerance, VMs.

List of Acronyms

ARIMA	Auto Regressive Integrated Moving Average
AFDS	Adaptive Fault Tolerance Dynamic Strategy
DB	Database
FT	Fault Tolerance
GB	Giga Byte
GZ	GigaHertz
IDE	Integrated Development Environment
LB	Load Balancing
LVQ	Learning vector quantization
MS	Microsoft
NIST	National Institute of Standards and Technology
OCI	Optimal Checkpoint Algorithm
OPVMP	Optimal Redundant Virtual Machine Placement
OS	Operating System
QoS	Quality of Service
SVM	Support Vector Machine
U. S	United States
VM	Virtual Machine
SPSS	Statistical Package for the Social Sciences
IBM	International Business Machines
TBAHFT	Threshold Based Adaptive Hybrid Fault Tolerance

LAN	Large Area Network
WAN	Wide Area Network
AWS	Amazon Web Services
SaaS	Software as Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
PLC	Programmable Logic Controller
MLAG	Multi-Chassis Link Aggregation
SLA	Service Level Agreement
CCaaS	Component-based Cloud Computing Systems
OCA	Optimal Checkpoint Algorithm
CSP	Cloud Service Provider
DAFT	Dynamic Adaptive Fault Tolerance
FPM	Failure Predictor Module
NDTBAFT	Non-Distributed Threshold Based Adaptive Fault Tolerance
NoFT	No Fault Tolerance

Acknowledgment

First and foremost, I give thanks and praise to my Almighty God who has been with me every step of the way throughout this thesis-writing process. I thank Him for providing me with knowledge, strength, guidance and wisdom. I extend my profound gratitude to my advisor Dr-Ing Frezewd Lemma for guiding, nurturing and supporting me during the entire course of writing this paper. His unconditional sage advice is invaluable and certainly helped me stay focused and reach this milestone. Most importantly, I want to express heartfelt appreciation to my mother, Gudetu Lechisa for her ongoing prayers throughout this process. I would like my family for their unwavering support and their contribution towards the success of this project couldn't go unnoticed. Finally, I am grateful beyond words for my dearest few friends that have always had a special place in my heart for their love and emotional support throughout this journey. Their genuine support meant so much at any point in time. God bless y'all!

CHAPTER ONE

1. Introduction

1.1. Background of the study

Cloud computing refers to accessing, configuring, and manipulating resources (such as software and hardware) at a remote location (Kumari & Kaur, 2021). (Buyya, Yeo, et al., 2009) defined Cloud computing in terms of distributed computing ``A Cloud is a type of parallel and distributed system containing a set of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers".

According to the U.S. National Institute of Standards and Technology (NIST) definition: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (for example servers, networks, storage, services, and applications) that can be quickly provisioned and released with least management effort or service provider interaction" (Osorio et al., 2006).

Cloud computing offers various resources in the form of services to the end users on-demand basis. It enables businesses and users to use applications without installing them on physical machines and allows access to required resources over the Internet. Delivering features such as high performance, pay-as-you-go, connectivity, interactivity, reliability, ease of programming, efficiency, scalability, big data management, and elasticity to transform IT from products to services (Savu, 2011).

Today's businesses use cloud computing systems because cloud applications provide functionality. To realize a good way of working, it is necessary to ensure the quality of business services. It discusses quality and its factors, quality of service QoS, cloud computing, advantages and disadvantages, and architectures and techniques used to provide QoS to cloud applications. By definition, quality should refer to requirements to imply that they must exist.(Goundar et al., 2018)

Therefore, there are no requirements and no quality (Elshaer, 2012). Many factors affect the quality of systems and applications. Flexibility is the software's ability to manage functionality without breaking the system. Maintainability, readability, and maintainability are a bit like flexibility, but with more emphasis on bug-fixing changes performance and efficiency.

Performance refers to system response time. Scalability a scalable system responds to the user's actions within an acceptable amount of time availability and robustness. A robust system should be available even when errors occur. Ease of use and accessibility. The user interface is the part of the software that has presented to the user, so it should be easy to use. Platform Compatibility a quality system should run on as many different platforms as possible. Therefore, it is intended for many users of the system or software (Amin et al., 2015).

Quality of service generally refers to the ability of a network to achieve maximum bandwidth and handle other network factors such as latency, error rate, and uptime. Quality of Service includes the management of other network resources by allocating priorities to specific types of data (audio, video, and file).

The basic implementation of QoS requires three main components: QoS within network elements, QoS policy and management capabilities to control end-to-end traffic on the network, and identification techniques for coordinating end-to-end QoS between network elements (Lutkevich, n.d.).

In the area of quality of service (QoS) management, the cloud creates new topics. QoS areas represent cloud-like areas of reliability, performance, platform, and availability provided by specific applications and hosts (Ganesh et al., 2014). With advances in cloud computing, QoS properties are constantly in the spotlight. Significant disruptions in QoS prediction, assurance, and analysis occur due to the heterogeneity and resource isolation mechanisms of cloud platforms. In cloud computing, fault tolerance is the ability of the cloud to withstand unexpected changes caused by hardware failures, software failures, network congestion, and so on. There are mainly two standard fault-tolerant policies available for real-time applications hosted in the cloud namely Proactive Fault Tolerance Policy and Reactive Fault Tolerant Policy (Ganga & Karthik, 2013).

Proactive Fault Tolerance: The principle of proactive fault tolerance is to avoid failures and recover from failures, anticipate failures, and proactively replace suspicious components with other running components. Some of the techniques based on proactive fault tolerance policies are:

Preemptive Migration: This is a method that makes use of a remarks loop to manipulate gadgets wherein the utility is continuously monitored and analyzed.

Software Rejuvenation: This is associated with the normal reboot agenda of the gadget. Each time you reboot, your gadget will remain clean.

Reactive Fault Tolerance: Reactive Fault Tolerance guidelines relate to measures that might be carried out to mitigate the effect of mistakes that have already befell inside the cloud. There are numerous strategies primarily based totally on retroactive fault tolerance guidelines, inclusive of test pointing, restarting, replication, and mission resubmission.

Checkpointing/Restart: In Checkpointing/Restart mechanism, the state of a system -that is running an application- is recorded in a global checkpoint. Therefore, in the event of an error, you can reset the system status to a checkpoint and continue from this point instead of starting the application from scratch.

Replication: The replication-based method is one of the most common fault tolerance methods. Replication is the process of keeping different copies of a data item or object on different resources. Replication adds redundancy to the system.

Task Resubmission: When an error is detected, the task is transferred to the same resource or another at run time without interrupting the system workflow.

The cloud computing model associates different technologies such as virtualization, virtual servers, data centers, and utility-based pricing and provides a new perspective that enables one to perform different operations more proficiently and cost-effectively. Cloud computing’s objective is to provide reliable services to all consumers transparently without unveiling the essential details of operations. Though cloud computing has several benefits, complexity, increased functionality, and cloud usage produce a high probability of resource failure.

1.2. Motivation

In cloud computing, the provision of a good service requires qualities of service parameters to be satisfied. Quality of service (QoS) is a measure of the overall performance of a service, such as reliability, availability, etc. Thus, with the huge growth of the Internet and its users, cloud computing has been guaranteed for both business and non-business computing customers by its incredible ease of use, quality of service, and management features of interest. It has become a computing platform. It is an adoptable technology as it provides integration of software and resources which are dynamically scalable. The dynamic environment of the cloud results in various unexpected faults and failures. In order to achieve robustness and dependability in cloud computing, failure should be assessed and handled effectively (Amin et al., 2015). The main motivation for designing a hybrid fault tolerance approach using proactive and reactive approaches is to provide higher availability and reliability than either of the individual techniques can provide alone. Proactive approaches, such as replicated components, deterministic algorithms, or self-checking mechanisms, can detect faults quickly; however, they do not provide any corrective action. On the other hand, reactive techniques such as failure masking schemes or fault recovery mechanisms allow recovering from errors in a timely manner; however, they have limited fault detecting capabilities. By combining the best of both worlds through a hybrid approach, it is possible to benefit from the high detection capabilities of proactive methods as well as fast corrective actions that reactive techniques offer. Therefore, one can increase system availability and reliability during operational environments, which can improve user satisfaction, productivity and reduce downtime costs.

1.3. Statement of the problem

In cloud computing, fault tolerance is a very important aspect in ensuring the quality-of-service provision. Consequently, fault tolerance in cloud computing is about designing a blueprint for continuing the ongoing work whenever a few parts are down or unavailable. Since cloud applications demand high availability, reduced execution time and reliability, the fault tolerance of the system or cloud applications can optimize the quality of the service in a cloud environment. Failure in cloud computing is inevitable. There are different fault tolerance methods by which different solutions for reduced execution time, high throughput and low failure rate have been addressed separately in the past few years. However, there is no particular fault tolerance approach with high throughput, execution time and low failure rate so that our cloud system is reliable and available. To address this it is important to combine the two approaches. Consequently, we can combine two fault tolerance approaches; namely proactive and reactive fault tolerance approaches to form a hybrid fault tolerance approach. Thus, a hybrid fault tolerance approach can handle both simultaneously to ensure the reliability of the cloud system.

1.4. Research Questions

This thesis intends to answer the following three research questions:

1. How can we design a hybrid fault tolerant cloud system using proactive and reactive approaches in a cloud computing environment?

2. What are the different parameters that can be used to determine the cloud environment as reliable?

3. How can we improve the parameters using a hybrid fault tolerance approach in a cloud environment?

1.5. General and specific objectives

1.5.1. General objective

The general objective of this study is to increase the reliability of the cloud computing environment by provisioning hybrid fault tolerance.

1.5.2. Specific objectives

- Analysis of existing fault tolerance technologies and techniques in cloud computing.
- Develop an algorithm to make cloud computing services that ensure reliability by improving execution time, throughput, and migration time.
- Identify a suitable simulator for implementation based on the requirement of the proposed task over a cloud environment.
- Implement the proposed hybrid fault tolerance algorithm using a simulation tool on the cloud to ensure the reliabilty of the cloud environment.
- Comparing the reliability of the cloud system using the proposed solution with previous works based on the parameters.

1.6. Significance of the study

This research is beneficial to increase the quality of services such as satisfying availability and reliability. Since this study will address both before and after a failure occurs, it would increase the quality of the service by ensuring fault tolerance over cloud computing. This work will be used to avoid service outages while accessing a given service from cloud-based service providers. To enhance service quality, this research work will utilize resources that exist on cloud and end devices to reduce latency, and meet customers' need to access the services by reducing optimal downtime of the system. To prevent disruptions arising from a single point of failure, ensuring the high availability and business continuity of mission-critical applications or services.

User Benefits:

Improved performance: Hybrid fault tolerance systems use both local and cloud resources to enhance the performance of user applications. This can also reduce hardware costs by distributing workloads across multiple nodes allowing processing resources to remain optimally allocated across data centers or even continents.

Increased resilience: Hybrid fault tolerant systems provide increased reliability and resiliency since they are able to make use of more than one type of redundancy for essential business operations, such as backup power systems, firewalls, etc. In the event of a failure in any one system, the other resources can still keep operations running smoothly with less downtime associated with it.

Scalability: By utilizing cloudized resources and local redundant infrastructure, providers are able to better scale capacity to match customer demand without buying additional physical hardware or making major changes to traditional server capabilities often associated with high resource utilization peaks or large storage requirements.

Reduced risk: The distributed approach used when implementing hybrid fault tolerant systems allows for providers to better manage risk by ensuring that possible failure points exist both locally and in the cloud so services are more reliable overall when times of peak demand or unexpected outages occur in either environment simultaneously.

Improved performance & efficiency: The combination of local and cloud-based resource utilization allows providers to optimize performance levels through quick response times, reduced latency issues for applications saving time and money in the process due their fault tolerance architecture not requiring as many specialized hardware components upfront.

Future researchers and academicians: The proposed algorithm can be used by researchers and academicians for further optimization. It is also used at the university level for research and as an academic reference. Provider Benefits:

1.7. Scope of the study

According to the research gap analyzed, a description of the different types of failures and their causes in a cloud computing environment has been identified. Accordingly, by exploring the basic fault-tolerance approach used in cloud computing environments, this particular study starts from analyzing the various fault-tolerance approaches proposed in the literatures written on cloud computing environment fault tolerance approaches. Consequently, it is necessary to implement autonomous fault tolerance using various parameters in a cloud environment. Accordingly, the scope of this study is combining proactive fault tolerance approaches and reactive fault tolerance approach to form a hybrid approach to increase throughput, execution time and migration time of the cloud environment and to decrease the failure rate of the cloud environment.

1.8. Limitations of the study

The study excludes live migration and energy consumption. Real-world deployment or implementation was not feasible. Thus, this research as entirely conducted using a Cloudsim simulator, Cloud Analyst and data evaluation technique apps only.

1.9. Methodology of the Study

In conducting this research, we thoroughly examined existing knowledge in this area. Book reviews, journal articles, research papers, topic reports, and other online documents were reviewed to gain insight into previously studied methods and approaches fault tolerance techniques in a cloud computing environments to achieve the above objectives and answer the research question.

1.9.1 Implementation and Design tools

NetBeans, a development environment, has been chosen as the best tools for implementing the proposed solution. NetBeans includes a Java Development Kit for Java developer, and as a result, the CloudSim framework package, which is built by Java, is integrated for simulation, as well as Edraw Max as a design tool. Edraw is a design tool that allows software developers to create diagrams such as flowcharts, workflows, and UML diagrams. In the research, this tool is used to design and draw diagrams such as flowcharts and workflows.

Hardware Requirements

The hardware requirements for conducting this research are listed below.

✓ Processor: Intel(R) Core (TM) i3-3120M CPU @ 2.50GHz.

✓ RAM: Installed memory (RAM) 8.00GB.

✓ System Type: x64 based processor.

✓ Hard Disk: 500 GB

1.10 Organization of the Thesis

The research consists of six main chapters; the rest of the research is organized as follows:

Chapter One discusses the introduction and background of the study, which this particular study lays its foundation on.

Chapter Two is the detailed literature review about fault tolerance algorithms in the cloud computing environment. This section includes the architecture of cloud computing and fault tolerance algorithms, different types of t fault tolerance algorithms for improved quality of service in the cloud, and some related and recent research works.

Chapter Three presents the methodology of the research and strategy to identify fault tolerance algorithms solutions. This section describes the research design, data collection technique, sampling techniques, data sources, cloud computing tools, and issues of reliability and validity.

Chapter Four presents the proposed solution using the proposed algorithm in the cloud computing environment.

Chapter Five presents the experimental design and implementation of the proposed algorithm, the simulation configuration, and the experimental results of the proposed algorithm and existing algorithm and at last the conclusions and future works put forward.

CHAPTER TWO

2. Literature Review and Related Works

2.1 Literature Review

Cloud computing is characterized as a model for empowering omnipresent, helpful, on-request network access to a common pool of configurable figuring assets that can be quickly provisioned and delivered with negligible administration action or specialist organization connection. These norms go about as a guide that contains five fundamental characteristics, three service models, and four deployment models(Khan, 2019).

2.1.2 Overview of Cloud Computing

According to (Vuyyuru et al., 2012), cloud computing is a system that allows users to pay as they go for computing resources such as applications, storage, and processing power over the internet. Its goal is to allow users to profit from shared pools of configurable resources without having to worry about the low-level structure of the system. Instead, they can focus on their business problems.

Cloud computing is progressing constantly as a new type of computing. Cloud computing provides a way to use and access multiple server-based computational resources via a digital network internet connection using www. Cloud users can access the server resources using a computer, netbook, pad computer, smartphone, or other devices. Cloud computing applications are managed and provided by the cloud server.(Khan, 2019). In cloud computing, data is also stored remotely in the cloud configuration. Cloud computing provides a way to deliver computing resources over the internet. When we store pictures and videos online using webmail or a social networking site we use cloud computing services. Cloud computing makes IT management easier and more responsive and it is cost-effective for the changing needs of the business.

According to explanation (Mell & Grance, 2011), Cloud computing is an evolving concept that makes better use of multiple distributed resources that can be allocated to users as per requirements. This helps in cheaper and more efficient utilization of

available resources and easier handling of larger computational problems. Its advantages include transparency of resources, flexibility, location independence, reliability, affordability, and greater availability of services, etc. To provide these facilities, the tasks need to be scheduled properly on the resources so as to provide maximum performance in minimum time.

Cloud Computing Layout

Cloud computing is defined as a model for enabling omnipresent, helpful, on-demand network access to a shared pool of programmable figuring assets that can be instantly provided and delivered with minimal management activity or specialist organization connection. The five fundamental characteristics, three service models, and four deployment models included in these norms act as a guide.

Cloud computing is characterized as a model for empowering omnipresent, helpful, on-request network access to a common pool of configurable figuring assets that can be quickly provisioned and delivered with negligible administration action or specialist organization connection. These norms go about as a guide that contains five fundamental characteristics, three service models, and four deployment models. These standards act as a roadmap that encompasses five essential characteristics, three service models, and four deployment models of the cloud. The service delivered by the cloud is classified as Software as Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Cloud computing has gained great attention from both the scientific and manufacturing community because of its high-quality services with measurable balanced costs. According to the study of Leavitt, cloud computing resource usage showed great growth during successive years as it became more used by enterprises and organizations (Mohammad, 2018).

Cloud provides many facilities due to its vast area such as sharing of resources for different purposes. The primary aim of cloud computing is to give the most suitable access to remote scattered resources.

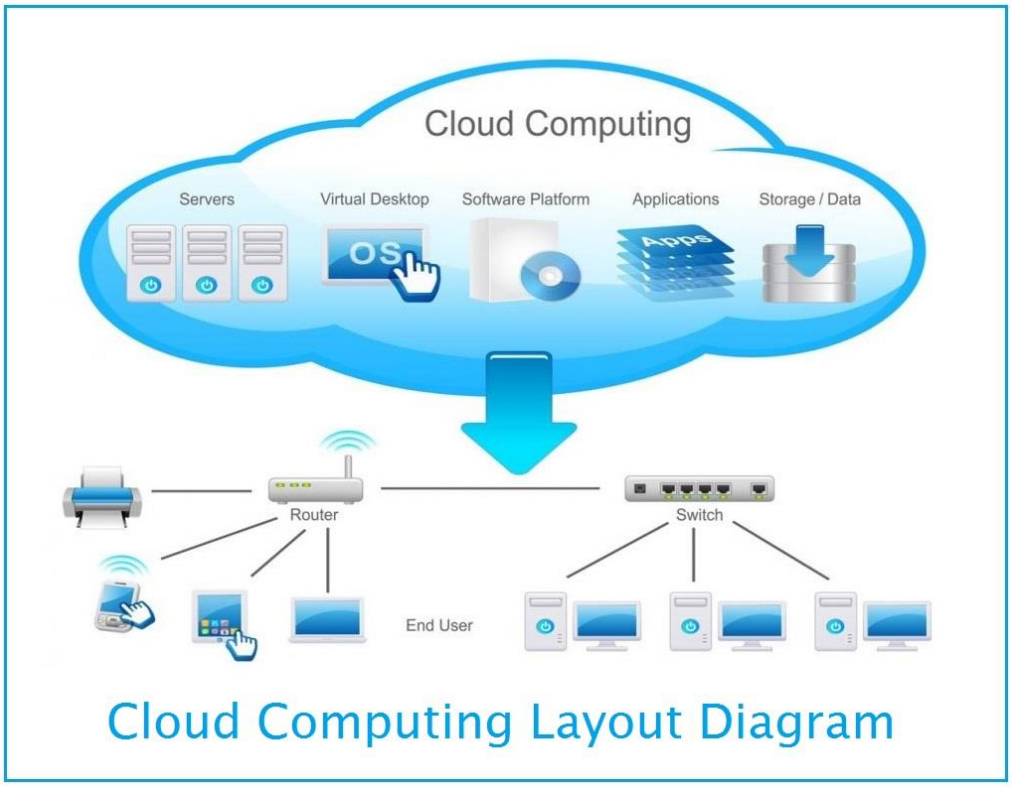

Figure 2. 1: Cloud Computing Layout((INFORAMTIONQ.com, n.d.)*)*

According to the explanation(Buyya, Ranjan, et al., 2009), Cloud computing is considered the latest network infrastructure that supports the decentralization of computing. The main features of the Cloud are the possibilities for building applications and providing various services to the end user by virtualization on the internet. One of the foremost challenges in the field of cloud computing is the occurrence of failures or outages at different levels. Task scheduling problem concerns the dynamic distribution of the tasks over the cloud resources to achieve the best results

2.1.2 Cloud Computing Architecture

Cloud computing architecture typically consists of a front end and a back end connected by Internet or Intranet networks. The front end comprises client devices which can be thin clients, fat clients, or mobile devices. The clients need some interfaces and applications for accessing the cloud computing system. The back end consists of various servers and data storage systems. A central server is typically used for administering the cloud system which includes monitoring the overall traffic and fulfilling the client demands in an efficient manner. Cloud computing is a new form of distributed computing that delivers infrastructure, platforms, software, and applications

as services. Cloud generally provides three levels of services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

Figure 2. 2: Cloud Computing Architecture(Nazari Cheraghlou et al., 2016)

2.1.3 Characteristics of Cloud Computing

For computing capability to be qualified as a service, it must have five major characteristics as discussed in NIST. Each provision of computing capability needs to be available on-demand, accessible through the internet, multitenant resources, and elastically provided.

1. On-Demand Services

On-demand computing service is a service model that provides every computing service based on the user's resource needs. Every service is made accessible on an "as needed" basis.

Accordingly, cloud hosting companies need to deliver computing resources to their clients when resources are required to get their customer satisfaction and reduce unwanted wastage of resources. Cloud hosting business companies have to easily scale resources to become valuable on changing market demand. A consumer needs to have the privilege solely to interact with computing capability without necessarily human intervention (Mell & Grance, 2011).

2. Broad Network Access

People need to access information existing anywhere by any device they possess either thin client or thick client platforms such as mobile phones, tablets, laptops, desktops, and workstations. One characteristic of cloud computing is to make its service available over a network and deliver it to the user through heterogeneous platforms. Customers are supposed to get information from cloud computing hosting companies irrespective of their current location and the device they are using.

3. Resource Pooling

Multiple customers could be served by sharing pooled resources which leads to low costs as a result of multitenant models. The multi-tenant model is an architecture that provides the ability to serve multiple users with a single computing resource but acts as a logically independent resource to each user. Resources can be storage, processing, memory, and network bandwidth that the customer has no knowledge of their exact locations but can identify the country or state they are available as a high-level abstraction.

4. Rapid Elasticity

Computing capability is provided in some cases automatically according to variable user demands. Dynamically allocating resources is to either shrink or grow computing capabilities based on consumer demand which enables it to appear to be unlimited. In older systems lack of resources such as storage and memory are often easily detectable to the consumer when their capacity reaches the maximum limit.

5. Measured Service

Transparency of resource past usage by the user needs to be apparent for both the resource suppliers and consumer based on metered services. Utilized services can be measured so that consumers have to pay for only services they use which in turn permits the supplier to automatically optimize and scale its resources based on needs. NIST stated that resources that are in use need to be transparent, visible, and measurable to both the resource provider and user of the resource so that the customer pays for the only resource they use (Mell & Grance, 2011).

Cloud Service Delivery Model

Cloud services are services that will be provisioned to customers over the internet allowing easy access to applications, storage, and processing resources without deep knowledge of internal infrastructure. Services given by the cloud are more helpful in reducing cost and scaling resources with more flexibility to provide it only on demands. NIST cloud computing definition defines three possible categories of services PaaS, SaaS, and IaaS(Mell & Grance, 2011).

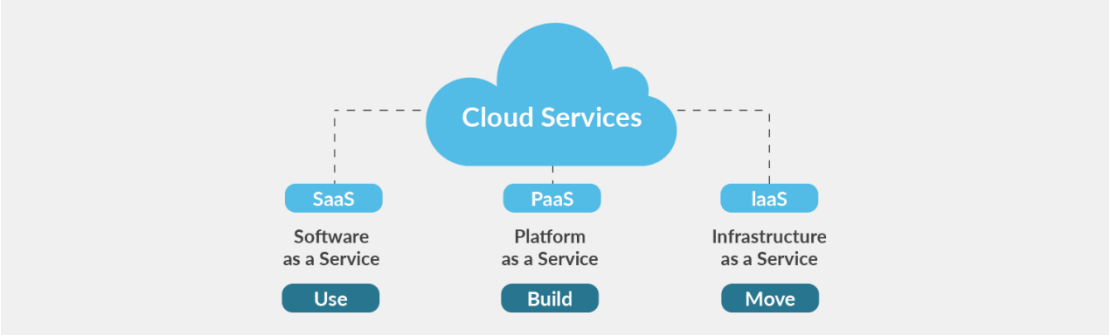

Figure 2. 3: Cloud Services (Mell & Grance, 2011)

1. Infrastructure as a Service

Infrastructure as a service is a cloud computing offering the vendor affords a customer access to hardware and software resources that can be storage, networking, servers, and other supporting resources including operating systems which afterward customers run their application and platforms within provided infrastructure (Mell & Grance, 2011). These allow the client to pay only for infrastructure such as processing and storage that might be scalable depending on demands. Therefore, users need not be concerned about the cost of buying and maintaining hardware. The user can monitor, manage, deploy and run arbitrary software which can embrace operating systems and other applications but they do not have control over the underlying cloud infrastructure.(Gibson et al., 2012)

2. Software as a Service

Software as a Service(SaaS) allows the customer to use software applications that reside remotely over cloud infrastructure which means users do not install, configure and upgrade the application on their local devices but access it through a web browser(e.g. web-based Email)or an API. This removes the cost of hardware purchase, provisioning,

and repairs, as well as software configuration, installation, and deployment (Gibson et al., 2012).

3. Platform as a Service

In this type of cloud service, cloud providers provide everything that developers need to build an application such as development tools, operating systems, and other infrastructures. All necessary management of tools for development will take place behind the scenes at cloud providers so that developers do not bother to update, install and configure the development environment (Gibson et al., 2012).

The services offered by PaaS to its users are an operating system, database management system, server software, storage, network access, hosting, and tools for design and development (Gibson et al., 2012). Those services are hosted by the cloud and users can access them through the web browser. Moreover, the user can select only the feature they need while rejecting other features that they do not require to include in their applications from preconfigured features given by the provider.

Cloud architecture can simply described as (Kastouni & Ait Lahcen, 2022)

Infrastructure-as-a-Service (IaaS): This layer provides the computing infrastructure, such as virtual machines and virtual networking, required for running an application or service.

Platform-as-a-Service (PaaS): This layer provides the run-time environment to develop applications and services. It also provides application development tools, middleware and other runtime components.

Software-as-a-Service (SaaS): This layer deploys applications and services with ready-to-use components that are specifically designed for Hybrid Fault Tolerance in Clouds. The TBAHFT Algorithm can be implemented by having a threshold on the failure parameters at this layer.

Database Service: This layer provides a mechanism to store data with transparency so the SaaS can access and use them. This layer is critical for storing information related to the fault tolerance performance of applications running in a cloud computing environment.

Networking Services: This encompasses all layers of network functionality including physical switches, routers, firewalls, load balancers, etc so that communication is reliable between different cloud components as well as users accessing the cloud environment from remote locations over the internet or Intranet networks.

2.1.4 Cloud Deployment Model

Cloud deployment model describes how it operates and who gets access to hosted cloud resources depending on user needs that require a reduction in their capital expenditure. The deployment model represents the category of cloud environment based on user requirement, size, and access as well as describes cloud purpose and nature. There are four types of cloud deployment models such as private cloud for a private organization, public cloud for general users, hybrid cloud, and community cloud [33].

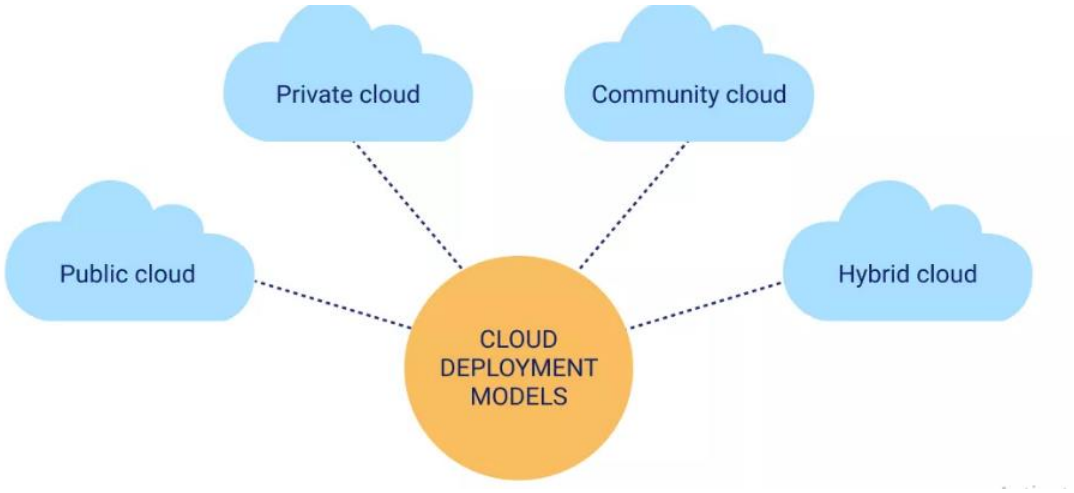

Figure 2. 4: Cloud deployment models

1. Private Cloud

According to(Liu et al., n.d.), this type of deployment model allows access to cloud resources only to specific organizations. In a sense, the cloud resource will be guarded by advanced firewalls under the investigation of the IT department that belongs to a specific organization.

Cloud resources and infrastructures are exclusively given to a single organization and solely managed by itself. Cloud resources can be hosted at organization premises (i.e. on-site private clouds) or at hosting companies (i.e. at cloud providers). Besides that, one benefit of using a private cloud for the organization is that it provides security and more control over its resource within the boundary of a privileged organization. A

private cloud can be deployed at the organization site which in that case higher performance is achieved as a result of the short distance of communication. In contrast, cost and logistical challenges associated with the management of required hardware could lead to a challenging problem(Liu et al., n.d.).

2. Public Cloud

(Kastouni & Ait Lahcen, 2022)The public cloud deployment model avails its cloud infrastructure to the general public and is possessed by an organization that vends services. Resource scale on demand can be realized and offered over an internet connection which will be paid based on a pay-per-usage fee. Additionally, the customer through a web browser or other software application uses services. The resource will be hosted at the cloud provider's site which manages and pools resources according to the capacity needed by users. Most importantly, the advantages of the public cloud include easy and inexpensive setup, on-demand scalability, full technician expertise, and data availability. While in contrast, public clouds are exposed to security and privacy issues since a third party manages resources. Some examples of public clouds are Google Gmail, Amazon AWS, and Microsoft Office 365.

3. Community Cloud

A community cloud works where two or more cloud customers have common objectives such as security, privacy, and mission objectives that need to share information. In the same way as other cloud models, it can be controlled by an organization or third party, or a combination of them. Besides this, it could be off-premises or on-premises depending on the simulators and tools(Liu et al., n.d.).

4. Hybrid Cloud

A hybrid deployment model is the combination of two or more other deployment models such as the private deployment model, public model, and community model. Most of the time it is a combination of the public cloud and private cloud. Through a hybrid deployment model, services and data from the other model can be integrated for achieving a unified, automated, and secured environment.(Liu et al., n.d.; Mell & Grance, 2011)

2.1.5 Fault tolerance in Cloud Computing

The fault tolerance technique is a technique of failover that enables the system to continue to work even if the failure occurred. The main role of the fault tolerance technique is to detect any fault that occurs in any part of the device, such as memory, disk space, network, or processing unit (CPU) (Sivagami & Easwarakumar, 2015). The purpose of the methodology of fault tolerance is to keep the system up and running with high system stability and reliability. Cloud computing providers are seeking to improve their platform by applying a fault tolerance strategy to achieve a high availability system, based on the importance of fault tolerance in cloud computing systems. The failures in cloud computing can be divided into two categories. First, software vulnerabilities such as abuse of information and incomplete data from the source. Second, failure of hardware like faulty or sluggish VMs and the exception of storage access (Alqahtani & Arishi, n.d.; R, 2015).

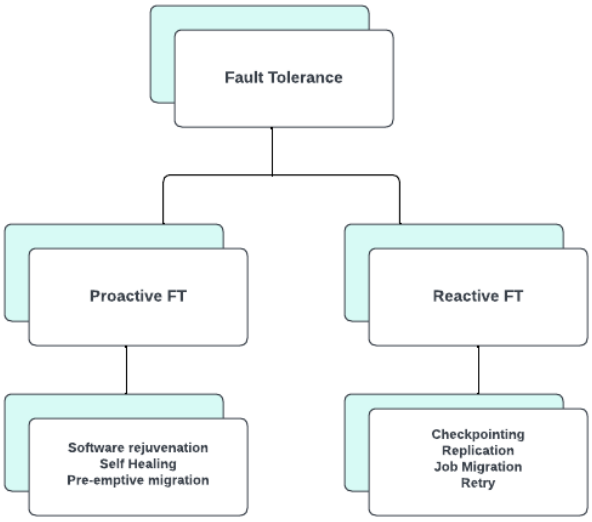

Figure 2. 5: Fault Tolerances in Cloud Computing

Conceptual Overview of Hybrid Fault Tolerant Algorithm

A hybrid fault tolerant algorithm should be able to provide an ideal combination of features from both VM migration and checkpointing strategies. In order to implement this approach, the faulty node must first be identified. Once the faulty node is identified, the concept of load balancing should be adopted in order to avoid any single virtual machine receiving too much load. Next, after a suitable balance has been established between the machines in terms of processing power, a cold (non-running) copy of all

data stored must be transferred from one machine to another corresponding duplicate copy on another healthy machine as part of the VM migration process. The data transfer should involve full transfers such as hardcopy along with data files as well as softcopy over any mobile medium such as the internet or LAN/WAN connection that links together both machines A and B that eventually makes up the original set (A+B). This cold copy will act as backup storage prior to initiating any checkpointing process at both source nodes A & source nodes B respectively making an effective virtual switch over taking place at instance t1 event ahead when necessary.

At subsequent time instants checkpoints or data snapshots need to be taken at regular intervals based on user defined policies either discreetly (at constant time-intervals) or continuously loitering about a predetermined stage around t1 utilizing upon existing network mechanisms; i.e. socket-programming etc.

Finally, these newly acquired checkpoints proceed next towards being moved from dormant nodes down onto live target nodes under consideration after successful verification checks taking place such activation criteria, etc. Ensuring no discrepant behavior emerging outwards at every instance along tracing program control flow graphs mapping each Execution Phase's entrance & exit points via previously aforementioned preparation step carried through earlier almost anytime now between M1-M4 instances thus creating a self regenerating. Furthermore, conceptually speaking all previously mentioned resilience enhancing strategies get effectively integrated into production code snip bits or prepare setup sheets whenever possible; allowing us to decentralize high availability feature right away decreasing response time penalty even further gains private network's efficiency rate: While also offering simultaneous multipath migrations rather than intervening restrictive single port model alone ultimately unlocking efficient utilization rate whilst reducing total energy consumption various delay/detection deadlock scenarios leading competitive system reliability survivability values driving long term endurance.

Implementing a hybrid fault tolerant algorithm requires coupling VM migration techniques with checkpointing techniques in order to build an adaptable system that is designed for failure avoidance, detection and management protocols therein lying beyond scope what forms all constitutive parts apart; namely fully fledged reporting, remote monitoring & database synchronization structures aptly conforming complimentary requirements during implementation going forward within reasonable amounts attained during operability induced testing stages rehearsed similarly weighted

point as usual answer precisely dependent case sample above open though individualized preambles up to furnish diverse range features customized ones geared "towards your advantage" where possible ahead parallelism market assesses continual progress gains organization houses responding promptly remark inclusive realistic predictions prove cumulative. The design for a hybrid fault tolerant algorithm by combining VM migration and checkpointing.

Proactive Fault Tolerance

Proactive fault tolerance in cloud computing is a design strategy aimed at ensuring systems availability. It works through the implementation of software techniques to identify and prevent predictable faults before they occur. In practical terms, proactive fault tolerance methods aim to detect abnormalities in system components or events, diagnose the cause of potential problems, and enact preventive measures to reduce or eliminate recurring issues (Alqahtani & Arishi, n.d.). This approach is enabled by a range of technologies such as machine learning, artificial intelligence (AI), autonomic decision making systems, and programmable logic controllers (PLCs). The main objectives of proactive fault tolerance strategies are improved system performance, enhanced reliability and availability, reduced repair costs and optimized system maintenance times. Proactive fault tolerance has become an increasingly important component of cloud computing due to its ability to improve operational efficiency while reducing risks associated with unpredictable system failure (Mesbahi et al., 2018).

Reactive fault tolerance in cloud computing is designed as a response technique that generally comes into effect after an issue has already occurred; reactive approaches attempt to mitigate the effects of faults rather than preventing them from occurring. Reactive strategies often involve monitoring processes for anomalous behavior or using redundancy constructs such as failover clusters or backup copies of data that can be called upon should an outage occur (Hasan & Goraya, 2018). In contrast with proactive measures which must seek out potential problems ahead of time, reactive strategies are able to address existing faults immediately when they are detected; this makes them useful for systems that require high levels of operational continuity or responsiveness from even disruptive events like power outages.

Overall, both proactive and reactive approaches offer significant benefits for users looking for robust fault protection options for their cloud environments. While reactive approaches are often seen as the less sophisticated methodology in comparison to more

comprehensive preventive efforts, their ‘firefighter’ mentality makes them essential in certain situations where downtime cannot be tolerated due costly related repercussions on the business environment (Hasan & Goraya, 2018). Ultimately both types of strategies have their place in driving uptime and availability, though it is often far simpler and faster to prevent issues from occurring than reactively addressing them after they arise.

Proactive fault tolerance is a type of techniques used to detect and protect against potential failure in order to prevent it from taking place. This technique focuses on anticipating and avoiding most problems before they even begin. Proactive fault tolerance can be achieved through proactive planned maintenance, problem prevention methods like systems backup, records tracking, and forecasting with predictive analytics. Furthermore, proactive strategies seek to identify issues before they create disruptions by relying on established processes, constant reviews and proper communication channels (Mavromatis & Pouloudi, 2012).

Reactive Fault Tolerance

Reactive fault tolerance is mainly concerned with responding quickly once an issue has been detected. This approach usually involves developing response plans that can address system breakdowns or outages as soon as they arise, minimizing downtime and the resulting economic impact (Kumar & Singh, 2017).

Reactive fault tolerance measures employ various strategies including redundant resources such as hardware components for when individuals parts malfunction or fail; fast remediation procedures to quickly identify root causes of problems; rolling updates for software applications that can restore functionality after a crash; incident escalation procedures for sending notification messages whether or not a problem is immediately addressed; and disaster preparedness plans that detail procedures users need to take (Huang et al., 2016).

Overall, both proactive and reactive approaches are required for effective fault tolerance techniques. While the proactive strategy focuses more on preparation in order to prevent issues from arising in the first place—or at least minimize their probability—the reactive measures focus on responding swiftly once any problem has been detected.

These complementary techniques work together in order to ensure that businesses remain safe when problems do occur (Huang et al., 2016).

Fault Tolerance Requirements

An important goal in designing distributed systems is constructing the system in such a way that it can automatically recover from partial failures without seriously affecting the overall performance. Fault tolerance is an important concern in any cloud computing environment. Fault tolerance can be defined as a system's ability to continue operating properly even when components are not functioning correctly (Rajabi, Pourhoseini, Aghabagheri, Habibi & Noorhosseini, 2018). Fault tolerance is crucial for achieving successful cloud computing systems as it helps ensure continuity despite any potential hardware or network failures. The requirements of fault tolerance involve multiple facets and must be carefully considered and implemented for a successful cloud solution.

The first part of fault tolerance in a cloud system involves the physical layer, which encompasses server availability. An effective Fault Tolerance Plan should have built-in redundancies that allow for seamless transition if one server fails (Kotenko, 2016). For example, resources like load balancers and redundant nodes should be included to increase overall resource availability and reduce downtime. Additionally, distributed data centers across different geographical locations also need to be factored into fault tolerance plans as they provide greater reliability and redundancy in case of natural disasters or other disruptions.

Network redundancy is another key factor in ensuring optimal fault-tolerance in any cloud platform. Network redundancies such as switch health monitoring or two separate uplinks serve the purpose of making sure that service is still available if one connection fail (Zaharia & Ghitulescu, 2017). Additional protocols such as MLAG (multi-chassis link aggregation) can also provide improved performance for high critical services by removing single points of failure from the network topology. Network redundancy helps ensure maximum uptime during an unexpected disaster situation or simply with scheduled maintenance tasks.

Reliability

Reliability is an important factor in achieving fault tolerance with a cloud system. Reliability refers to the dependability of a cloud infrastructure's components and services, including their ability to meet customer expectations. Cloud providers typically offer service level agreements (SLAs) to guarantee reliable performance and reimburse customers for any downtime that might occur due to failure. Achieving high reliability requires deploying redundant systems, using resilient components such as disks and power supplies, implementing high-availability architectures, and so forth (Li et al., 2017).

In order to achieve fault tolerance in a cloud system, reliability must be taken into consideration from the system's design phase. The goal is to design a system which maintains service availability even when any single component or individual hardware failure occurs. This can be accomplished by having multiple components working together such as redundant power switches and multiple network connections distributing the load between them (Cheung et al., 2018). These types of infrastructure configurations are known as server clusters or distributed systems, which present advantages over traditional models such as better scalability and higher resistance against outages and failures.

Additionally, fault-tolerant software must also be developed in order to ensure more reliable performance. Couromulles et al. (2018) suggest testing applications before deployment by simulating various failure scenarios and identifying possible impacts on the application's behavior while running under these conditions. Fault-tolerant software can be developed using rigorous testing processes during the design stage that help detect any unexpected malfunctions before they cause disruption in production environments (Jalali et al., 2019). The development should also include intrinsic functions that handle certain types of errors, stop processes if necessary, identify failed nodes, repair faults where possible, manage workloads among available nodes efficiently, protect data integrity despite intermittent failures, etc.

Throughput in cloud computing is the amount of data a user or application is able to transfer over a given time period in a cloud computing environment. Throughput is usually measured in megabits per second (Mbps) or gigabits per second (Gbps). The higher the throughput, the greater the speed of data transfer. Throughput can be

increased by adding more bandwidth and resources, similar to increasing download speeds for broadband connections(Rawat, Sushil, Agarwal, Sikander, et al., 2021).

2.2 Related Works

Nowadays, quality of service (QoS) is a trend in virtual networks to get the most out of resources. Computation models with high-quality service (QoS) have been proposed, and the requirements from service users are increasing. Work has been done in the areas of computing in a variety of areas, including service level agreements, server optimization, scientific workflow execution, resource management, allocation, and load balancing. Cloud computing environment faults that appear as failures to the end users can be categorized into two types similar to other distributed systems (Dhinesh Babu & Venkata Krishna, 2013).

A failure is a failure or condition that achieves the intended function or behavior of the system. There may be many reasons behind a failure that may happen due to various reasons that are due as reaching an invalid system state, network failure, etc. The belief that causes an error is a fault that represents a fundamental impairment in the system. Thus, Fault tolerance is the ability of the system to perform its function even in the presence of failures. It serves as one of the means to improve the overall system's dependability. In particular, it contributes significantly to increasing the system's reliability and availability.

(Sun et al., 2013) proposed a novel dynamic adaptive fault tolerance strategy which can improve the reliability of Cloud Computing. The proposed method consists of three stages: system monitoring, error handling and fault recovery. In the first stage, system monitoring is used to detect any sudden deterioration of quality-of-service (QoS) for timely response. The second stage is about error detection and handling which helps identify errors in QoS parameters to mitigate them quickly. Finally, the third stage involves the recovery process from multiple errors and faults by using the appropriate restore strategy.

(Khan et al., 2018) have proposed a novel approach for robustness improvement of component-based cloud computing systems. Component-based Cloud Computing Systems (CCaaS), involve the use of components such as storage and processing units that are used to manage the workload of the system. The objective of their work is to

tackle the problem of robustness in these systems while also increasing dependability and scalability. To this end, they have proposed the implementation of a secure gateway which would act as an intermediary between clients/application needs and cloud infrastructure. This gateway monitors communication and context information while enforcing security measures during request processing. Furthermore, they have also suggested an improved monitoring policy which tries to prevent malicious requests from being served by the system. This policy leverages machine learning algorithms to detect anomalies or suspicious activity in real time.

Additionally, their approach has three main approaches for improving system robustness: Trustworthiness metric calculation for each component; Usage & Input data trust assessment; and Utilization metrics & access control detection among different components. In order to evaluate the proposed approach, they conducted experiments using two sets of datasets; one from Microsoft Azure Virtual Machine Datacenter and another set from Google App Engine Datacenter. Results showed that trustworthiness metric calculation process achieved accuracy up to 98%. Their auditing solution consistently detected 81% of all malicious activities with false positive rate lower than 10%. Finally, their decentralized mode provide increased availability across clouds and already was proven more robust with latency reduction up to 70%. The proposed system can be extended for integration with Big Data frameworks such as MongoDB or Apache Spark Integration Systems (Khan et al., 2018).

(Rawat, Sushil, Agarwal, Sikander, et al., 2021) focused on developing an adaptive fault tolerance algorithm suitable for cloud computing environments. The authors create a new threshold-based algorithm that is not distributed and is designed to tolerate transient errors within the system while maintaining throughput efficiency. It applies the conventional Threshold-Based Fault Tolerance (TBFT) model to the context of cloud computing, however modified in a way to make it applicable in such an environment. The novel algorithm also includes two new components: dynamic reconfiguration techniques that determine how TBFT can react to changes in workload rate, and preemptive techniques that deactivate unnecessary instances when applications' workload reduces.

The research was implemented on two cloud infrastructures: Amazon EC2 and HPCloud, demonstrating high results against existing benchmarks and state-of-the art

models for similar cases. The algorithm successfully addressed issues concerning resource utilization and improved fault detection latency when compared with the traditional methods of distributed FT algorithms used in these environments. Furthermore, the authors discuss how the proposed model can be beneficial in improving other aspects of cloud performance such as scalability, availability and energy utilization. Their results concluded that this new approach can provide better fault tolerant measures while simultaneously reducing costs associated with running applications in public clouds(Rawat, Sushil, Agarwal, Sikander, et al., 2021).

(Zhang et al., 2018) presented an online fault detection model and strategies based on SVM-grid in clouds. This model uses the Support Vector Machine (SVM) algorithm to detect faults in the cloud environment. The SVM is extended using a fusion of grid technologies and optimized parameters are derived from the output. Cloud-based fault detection technology is obtained by applying synthetic datasets to run various experimentations and estimates. Finally, the effects of Parameter Gird Size, Learning Rate, Penalty Factor and Overlapping Ratio are discussed during the process. This result was also verified using large amount of failures data collected from real-world services like Google Photos. The results showed that proposed strategy was superior when compared to traditional methods, achieving better performance in terms of accuracy and stability in a timely manner. Consequently, this work proves to be an effective tool for fast and reliable fault detection system while minimizing the resource utilization at the same time.

Some reactive approaches had been proposed in the last few years some of the proposed models and frameworks based on reactive fault tolerance are:

(Zhou et al., 2017) proposed an OPVMP (optimal redundant virtual machine placement) model aims to optimize the placement of redundant virtual machines in order to maintain service continuity and reliability while minimizing costs. The model begins by constructing an application graph of a given system architecture, which is used to represent its set of components, relationships between components and availability requirements. Subsequently, the model proposes an objective function that considers redundancy information, execution context and preferences of users. This objective function is used to identify optimal server placements for redundant virtual machines in relation to other components to maintain service continuity. It also considers cost constraints imposed by users, including computing cost and availability requirements.

Finally, a heuristic algorithm is used for calculating server-allocation options which minimizes cost while maximizing system resource efficiency.

The proposed model has several advantages over existing approaches: firstly, it provides greater flexibility in terms of the selection of placement strategies; secondly, it allows users to specify their own criteria when making decisions; and finally, it offers improved functionalities such as scalability, elasticity and auto-scaling abilities with respect to future workloads or changes in system status. This helps them choose best suited optimization strategies based on their specific objectives and constraints. Additionally, the proposed solution yields optimal solutions that are able to reduce total costs without degrading performance or risking system reliability. Experimental results have shown that the proposed approach uses fewer network resources than other algorithms.

The work conducted by (Rawat, Sushil, Agarwal, Sikander, et al., 2021) focuses on FT techniques that follow stochastic prediction models to determine the failure of VMs based on historical states. (Rawat, Sushil, Agarwal, Sikander, et al., 2021) proposed work on the "Threshold-Based File Replication (TBFR)" FT model that replicates various responses from server configurations whenever a request exceeds the permissible threshold count. The work investigated by Liu et al. [23] who designed a framework based on the FT model which is preemptive and well coordinated for handling multiple VMs based applications. This methodology is supported by particle swarm optimization (PSO) for moderating such FT tolerance problems. The primary intention for implementing this model is to reduce overall consumption and transmission overhead for energy. In addition, for parallel applications, for well organizing the total execution time and handling network resources have also been briefed.

(Amoon, 2015) has developed a Framework for Providing a Hybrid Fault Tolerance in Cloud Computing, discusses a hybrid fault tolerance framework designed to ensure the reliability and availability of cloud computing applications. The proposed architecture is based on redundancy, replication, and recovery techniques to achieve fast restoration from failures with fewer resources. The authors outline a protocol model for their solution which includes four components: resource provisioning management, monitoring layer, replication layer, and data recovery layer. They address scalability

issues through their proposed framework by analyzing key trade-offs between layers such as periodicity of data backup operations and replica locations. Additionally, an analysis is conducted to evaluate the proposed framework's overall performance in comparison to existing solutions. The paper concludes by examining future work that can be done to further improve the proposed framework.

(Amoon, 2016) an Adaptive Framework for Reliable Cloud Computing Environment proposed a set of techniques to improve the reliability of cloud services. The proposed framework applied machine learning techniques, including unsupervised, semi-supervised and supervised learning. Moreover, it used two different types of neural networks – Convolutional Neural Networks (CNNs) and Long Short Term Memory network – for monitoring and detection purposes. Further, the paper used fuzzy logic based estimation methods for automatic prediction of resource utilization in order to estimate the reliability and performance of each service in the cloud. Lastly, it analyzed various rules and models such as entity-relationship models, traffic monitoring algorithms, behavior analysis model to find out any type of anomalies or violations that might occur in the cloud environment. The primary advantage of this framework is that it not only helps ensure reliable cloud computing environment but also reduces operational cost. The performance of the proposed framework was compared to the existing OCI (Optimal Checkpoint Algorithm). As a result, the proposed framework adaptability has improved the performance of cloud environments compared to existing algorithms.

(Abdelfattah et al., n.d.) Proposed hybrid approach which used reactive and proactive policies. The objective of this work was to use resources efficiently by reducing the consumption of these resources. It tries to create a resilient cloud computing environment by using a combination of replication-based fault tolerance and reactive intrusion detection techniques. According to the authors, this hybrid approach is able to ensure higher levels of availability, scalability and reliability for cloud services.

The work of (Abdelfattah et al., n.d.) applies the concept of replication impact heuristics in order to detect faults and intrusions in a highly dynamic manner. In addition, it uses reactive intrusion detection systems that rely on machine learning algorithms to continuously monitor and modify security configurations of a cloud service provider (CSP). The hybrid approach combines the inherent strengths and benefits of both

replication-based and reactive intrusion detection approaches in order to reduce system downtime while providing high levels of scalability, availability and security. Moreover, the authors also propose an automated mechanism for selecting suitable CSPs depending upon on an application's requirements and threats.

Overall, Hybrid Fault Tolerance Approach Based on Replication Impact Heuristic for Cloud Computing provides a useful framework that can be used by CSPs when developing their own architectures while still allowing customization. This novel hybrid method has the potential to bring more stability and reliability in cloud computing platforms compared to other traditional approaches. This approach enhances the fault tolerance by increasing the VMs efficiency based on the replication impact heuristic. Their experiments were evaluated by using a CloudSim simulator to execute tasks. A user defines the number of rent resources and several resubmissions of tasks. Then a replica heuristic defines a certain replica for each task. The task is executed at a certain VM, not all available VMs at the same time. If the current task passed to finish its job the next task is called to execute. But if a task fails to finish its job at VMs, it is resubmitted again to the best VM which has a high-reliability value. If the task fails again, resubmit it again until it exceeds the resubmission number. If they exceed the resubmission number a rejected message is sent to the client.(Abdelfattah et al., n.d.)

The work (Sun et al., 2013)proposed “dynamic adaptive fault tolerance (DAFT)” that utilizes the mathematical-based model which bridges the failure rates and its proposed two FT techniques on check-points and data replication process. DAFT is a novel approach towards addressing the issue of node failures in distributed systems, with potential for application in safety-critical devices.

By using localized information about near-node devices, DAFT enables robust execution despite dynamic changes to the system environment. The algorithm is designed to be modular and divided into four sections: failfast handling, monitoring, contingent recovery path planning, and quick start recovery. Through simulation analysis and experiments with ten different fault scenarios applied on a robotic swarm, DAFT demonstrated shorter convergence time and accurate decisions when compared to existing approaches. Furthermore, it coped well with dynamic fault scenarios which required self-adaptation, thereby proving its effectiveness against unpredictable disturbances or changes. In conclusion, the authors proposed an innovative system that

provides an efficient fault tolerance solution that can accommodate diverse distributed settings while ensuring system resilience and reliability(Sun et al., 2013). Availability is the measure of how quickly and consistently a service can be accessed or used. Availability's goal is to ensure that the service will be available when needed. Reliability is the measure of how suitable a service can be trusted with regards to accuracy, consistency, and dependability. A reliable cloud computing system should be able to provide a certain level of results despite changes in external conditions and also remain in working order even if one component fail

Title and Author	Objective	Methodology	Gap
(Abdelfattah et al., 2019) Hybrid Fault Tolerance Approach Based on Replication Impact Heuristic for Cloud Computing	The objective of this work was to use resources efficiently by reducing the consumption of these resources.	The proposed approach aims to improve the existing solutions by balancing the tradeoff between the two approaches. It tries to create a resilient cloud computing environment by using a combination of replication-based fault tolerance and reactive intrusion detection techniques. This hybrid approach is able to ensure higher levels of availability, scalability and reliability for cloud services. The CloudSim is a simulator that has been used to evaluate the experiments of four scenarios.	This work didn't improve failure rate and execution time.
(Nosrati, M.et al.,2018) NDTBAFT in cloud computing environments.	To ensure the effective and efficient execution of distributed applications in the presence of various faults.	Consists of implementing various strategies utilizing existing resources, assessing their effectiveness through continuous testing & evaluation with changing use cases rapidly adjusting them thanks to the dynamic nature of cloud services can help provide reliable connectivity whilst maintaining regulatory compliance where required	Low throughput and high migration time

Zhang et al. (2018) SVM-Grid online FD	It has provided improved accuracy and lower time cost as compared to BP, LVQ, and traditional SVM	FT algorithms for sample DBs (databases) have been developed. Simulation experiments have been done using a dataset of the Google corporation (Google2 application cluster).	This work can only predict the occurrence of faults in clouds, not how to ensure fault tolerance in the cloud in both approaches.
(Sheetal et al., 2021) Cost Effective Hybrid Fault Tolerant Scheduling Model for Cloud Computing Environment.	To develop a cost-effective hybrid fault-tolerant scheduling model for cloud computing environments it combines existing fault-tolerant techniques to improve cloud performance and increases their reliability while decreasing their costs.	The proposed model considers both intra-zone and inter-zone fault-tolerant mechanisms to address system failure challenges in the cloud computing environment.	High execution time and migration time during runtime in the cloud environment.

(Rawat, Sushil, Agarwal, & Sikander, 2021) A New Approach for VM Failure Prediction using Stochastic Model in Cloud	The objective of this work is to utilize historical system performance data along with machine learning techniques to improve the accuracy of failure prediction and minimize performance degradation	It starts by gathering requirements such as trustworthiness, high availability, scalability and services across multiple domains. They then created an architecture with two layers to provide fault tolerance and context based adaptation. The framework was evaluated through experiments and simulations to determine whether it meets these requirements	Migration time and execution time
(Rawat, Sushil, Agarwal, Sikander, et al., 2021) A New Adaptive Fault Tolerant Framework in the Cloud	It aims to identify potential fault occurrences early, prevent them from happening and quickly recover when they do occur, while minimizing resource utilization and data loss.	The framework proposes an adaptive fault tolerant mechanism which monitors, diagnoses and repairs cloud resources in real-time. It evaluates performance metrics such as availability and throughput to detect faults that might impact the service then finally it recommends an appropriate FT.	Failure rate issues were not addressed
(Zhou et al., 2017)Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization	To enhance service reliability in the cloud computing system	This work proposes a redundant VM placement optimization approach to enhancing the reliability of cloud services.	This work didn't address execution time, only in reactive approach

Zhang et al. (2018) SVM-Grid online FD	It has provided improved accuracy and lower time cost as compared to BP, LVQ, and traditional SVM	FT algorithms for sample DBs (databases) have been developed. Simulation experiments have been done using a dataset of the Google corporation (Google2 application cluster).	This work can only predict the occurrence of faults in clouds, not how to ensure fault tolerance in the cloud in both approaches.

CHAPTER THREE

3. Materials and Method

3.1 Methodology

In order to achieve both general and specific objectives of the research, different tools and techniques have been used. Fundamentally Computer Science research methodology has been be used in designing hybrid fault tolerance by combining proactive and reactive approaches through various steps. First, a systematic review has been conducted to identify relevant literature on proactive and reactive approaches to fault tolerance and their differences. This helps to provide an understanding of the key aspects of each approach that could be used for an effective combination. Second, system requirements was identified including reliability, scalability, cost, efficiency and any other specific requirements. Third, experiments and simulations can be carried out to understand the separate behaviour of proactive and reactive systems to assess how these components interact in a hybrid architecture and help design the best architecture for hybrid fault tolerance. Fourth, a simulation model selected by taking into account all the constraints defined during the previous steps. Fifth, empirical evaluation based on experiments or simulations should be carried out to validate the proposed model concerning different environmental conditions like load distribution behaviours etc. Lastly, a comprehensive analysis of results obtained from evaluation will help create guidelines for fine-tuning the hybrid Fault Tolerance system which includes recommendations regarding resource provisioning across different tiers of services or application level redundancy etc. Below are the detailed descriptions.

Systematic Literature Review

A systematic review of published papers, survey papers, books, and official websites has been conducted to have an understanding of fault tolerance techniques, challenges, and prospects that are associated.

3.2 Research Design

Research is described as an approach to promote knowledge enhancement. Research design is based on the concept of a designer trying to find a solution related to a human problem. The creation of innovative artifacts is an important part of the process, whereas the designed artifacts are useful and fundamental in understanding the problem. Artifacts are categorized into several components, including constructs, models (abstractions), methods (algorithms and practices), and instantiations (implemented and prototype systems).

Our research study focuses on the descriptive research type which systematically describes the problem, program, and characteristics of an individual situation. Two types of research methods: Qualitative research method and Quantitative research method. In this study, we used quantitative research methods that describe how a new algorithm is expected to support solutions to problems.

Quantitative research utilizes numerical analysis and tries to illustrate the relationship among issues in this study. Thus, quantitative research often deals with surveys or experiments, qualitative research can use procedures such as field studies.

Most of the time the original reports are found in academic journals. Additionally, the detailing of the research methodology used in-depth descriptions of the research. The activities of research design begin with problem formulation and move on with suggestions on how to solve it. Development and Evaluation are proceeding, with the possibility of moving back.

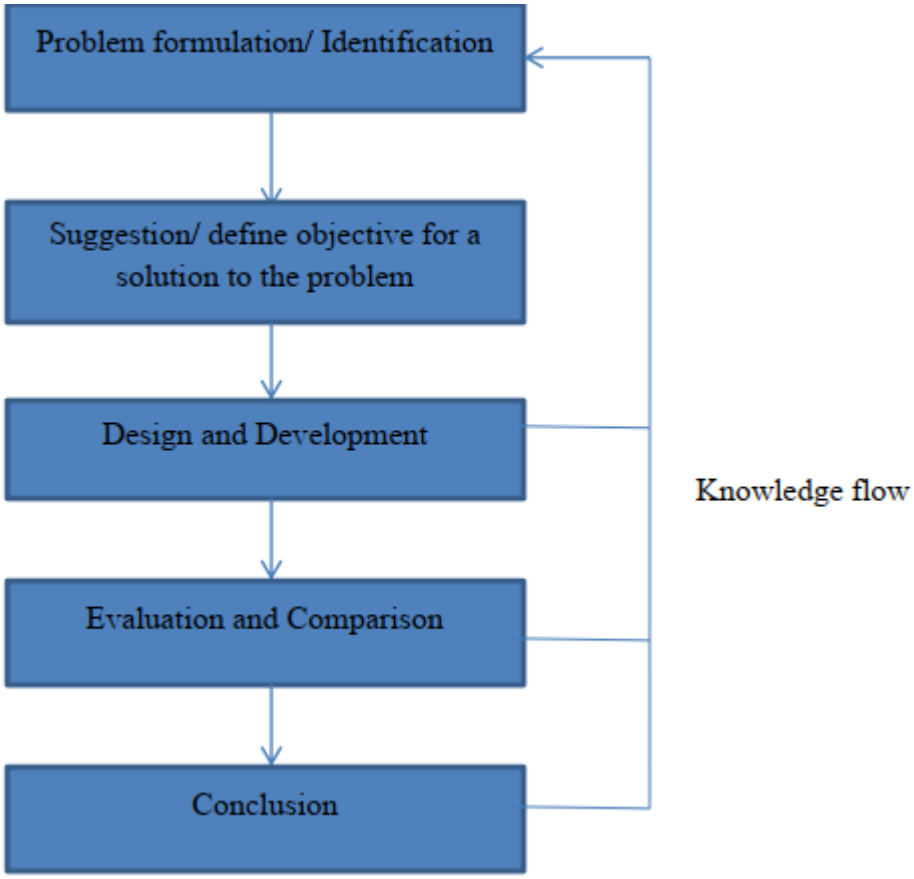

Figure 3. 1: Research cycle

3.2.1 Problem formulation / Identification

The problem identification was used as a requirement for the development of an algorithm. Reading similar and relevant research published in conference proceedings and journals are essential, which is presented in Chapter 2 (Literature Review).

Review of several algorithms which focus on fault tolerance in cloud computing. These reviewed algorithms have also been used in other research areas such as reactive SVR, grid Computing, distributed computing, etc. It is already known that the byzantine algorithm is a famous algorithm along with different proactive and reactive fault tolerance algorithms. Thus, I have tried to clearly identify the gap in the existing proactive and reactive algorithms by combining the two together and the parameter used for evaluation techniques.

3.2.2 Objective for solution

The objective of this study addresses the tasks needed to solve the problem of the research. Objective can be quantitative how the current one is better than the existing one how the new algorithm is expected to support a solution to the problem not yet addressed.

3.2.3 Design and Development

In this activity, the idea is to develop a unique architecture design, for extensibility, and modularity and define functionalities of system components and interrelationships among them. After determining the architecture of the new algorithm with its functionality, implement the proposed algorithm in cloud analyst toolkit which runs Eclipse IDE with a different number of independent tasks and different number of VMs.

3.2.4 Evaluation and Comparison

After understanding the simulation tool then implement the proposed scheduling algorithm and evaluate the performance based on different metrics such as response time and processing time. Make a comparison of the proposed algorithm performance with the basic hybrid algorithm. Analyzing and discussing the experimental result of the proposed hybrid fault tolerance algorithm and the existing fault tolerance algorithms is the core of this section.

3.3 Data Collection

For the study purpose, both primary and secondary data are used. In this research, mostly we used secondary data as data source for the study such as articles, journals, research, and other topics related to task scheduling.

Articles\ Full-text research from early 2015 to the current year 2022 were selected based on relevance to the research.

The sources were from the Google Scholar search engine and various research online databases. External sources such as the Mendeley database, and generated using simulation tools to develop and evaluate the proposed solution. Based on this

information, the effectiveness of the proposed solution will be assessed and compared to other existing approaches.

Primary data are the data collected by a researcher specifically for a research assignment. In other words, information that an organization must gather because no one has compiled and published the information in a forum accessible to the public. The primary data are unique and relevant to the topic of the research study so the degree of accuracy is very high. Moreover, primary data is current and it can better give a realistic view to the researcher about the topic under consideration.

On the other hand, Secondary data is the data collected by a party not related to the research study but collected these data for some other purpose and at different times in the past. In this research study, this data becomes secondary data for the current users. These may be available in written, typed or electronic forms. A variety of secondary information sources is available to gather data on the industry, potential product applications, and the marketplace. Secondary data is also used to gain an initial perception of the research problem and classified in terms of its source either internal or external.

3.4. Data Collection Techniques

The data collection is the gathering of data for a particular purpose from various sources and is used to explain the process of collecting data. The information or data can come from a range of sources. To collect the primary data, we use different types of techniques.

For instance, in this research, we used googling or action search techniques to collect data from the internet. Data collection is the process of gathering and measuring the information or data on objected variables in an established systematic fashion. In addition, it also enables a person or organization in order to answer relevant questions and evaluate the outcomes, and make predictions about future probabilities and trends. In this research study, data collection is defined as the systematic collection, analysis, and interpretation of the necessary data to design, implement and evaluate. It is also used to develop effective strategies.

3.5 Research Process

A threshold-based mechanism is implemented to guarantee cloud service availability and reliability in TBAHFT. The objective of the proposed approach is to monitor and anticipate the faulty VMs. When there is an occurrence of a faulty VM, the proposed approach will identify the spare for the VMs running on the failing host based on the threshold values. The major component of the TBAHFT architecture is AM, responsible for determining the most suitable FT action during VM failure. Designing an optimized AM is challenging. First, it must consider cloud application performance factors. Second, it includes the available spare nodes, costs, and benefits of different FT techniques. It consists of three significant components classifier, migrator, and replicator with some ratio for each objective resulting in a single objective which becomes the optimum solution for all objectives. As a result, it would be easy to get and evaluate a better solution on a single objective. Before scheduling a provider identifies a ratio for energy, and cost that must be considered, based on this scheduler finds the best solution out of possible solutions by multiplying both energy and cost with the ratio provided and adding the result to evaluate the solution with another solution found.

3.6 Issues of Reliability and Validity

Evaluating the quality of research is essential if findings are to be utilized in practice and incorporated into care delivery. Validity is the precision in which the findings accurately reflect the data. Reliability is the consistency of the analytical procedures including accounting for personal and research method biases that may have influenced the findings. Generalizability is the transferability of the findings to other settings and applicability in other contexts [21].

In this research, I used different validation criteria and compared the result in terms of overall response time, processing time, and others. The quality of research is defined depending on reliability and validity and measures the performance of a proposed threshold-based fault tolerance algorithm using the CloudSim tool.

3.7 Cloud Computing Tools

The proposed approach is tested using the Cloud Simulator tool which runs on eclipse IDE with a different number of independent tasks and a different number of VMs.

Experiment Environment and Tools

Various environments and tools have been used for the accomplishment of this particular work. Cloud computing simulation tools are virtual environments designed and developed to resemble physical environment features to reduce the cost of buying and configuring physical devices to develop and simulate research works.
The most commonly widely used simulators are CloudSim and CloudAnalyst.

CloudSim is an open source java-based simulation tool that contains different layers used for modeling, programming, and experimentation of cloud-based simulation. It is designed and developed in the GRID laboratory at Melbourne University [29]. CloudSim provides a lot of different policies of virtual machine allocation, a network connection within elements, flexible allocation of the processor core, and cloudlet management. In cloudsim, there are classes responsible for the provision of computational resources, virtual machine configuration, data center management, and user task management [30]. By using this simulator, researchers can design and develop a solution for cloud-related problems without wasting time and cost in configuring physical devices

I. Cloudsim

In Buyya's explanation, CloudSim is a framework developed by the GRIDS laboratory of University of Melbourne which enables seamless modeling, simulation and experimenting on designing cloud computing infrastructures. Cloudsim is a self-contained platform which can be used to model DCs, service brokers, scheduling and allocation policies of a large scaled cloud platform. It provides a virtualization engine with extensive structures for modeling the creation and life cycle management of virtual engines in a DC. CloudSim framework is constructed on top of gridsim framework also developed by the GRIDS laboratory(Buyya, Ranjan, et al., 2009)

3.8. Evaluation Approach

Based on the cloud analyst open source tool the hybrid fault tolerance algorithm is evaluated in different parameters such as availability, execution time, reliability, and others. In the existing fault tolerance algorithm when the number of cloud users increases to access cloud resources the complexity of the task scheduling algorithm also increases and there is energy consumption because of overloaded problems. But in the proposed fault tolerance algorithm if there is an overloaded virtual machine the task migrates to an underloaded virtual machine to minimize energy consumption.

Due to this we can reduce the complexity of the algorithm and increase the performance of the algorithm.

CHAPTER 4

4. Experiment Result and Discussion

Research Simulation Tools

Cloudsim enables researchers to focus on specific system design issues rather than worrying about low-level details associated with cloud-based infrastructures and services. It is a java based toolkit that includes actors or entities that are nothing more than java classes that communicate with each other during the simulation process (Humane & Varshapriya, 2015).

Cloudsim have the following common entity or actors which performed on action

1. **Virtual machine (VM)** is logical machine on which application or task will be run.

2. **Data center (DC)** this provisions the resources, and may have one or more than one hosts

3. **Host** is a physical machine on which logical machines are placed.

4. Cloud Information Services (CIS) an entity which manages all registering of resources

6. **Broker** is an agent, who will submit the VMs in specific host and then bind cloud lets, applications, processes on specific VMs and after execution of all the cloud-lets, the free VMs will be automatically destroyed.

7**. Data center Broker** an agent who is responsible for communications between data center and VMs as well as Cloud-lets

8. **Cloud-lets** are the task or application that run on VMs Proposed Solution

To go through the execute the proposed work Fault Tolerance Manager is used: To predict faults to increase the service reliability of Cloud computing. This model consists of various managers who have different workings which are divided into two phases:

- Storage of data Processing Storage of data is handled by two managers i.e. recovery manager and replication manager. The recovery manager takes care of two failures i.e. database failure and hardware failure. It helps in recovering the data if these failures occur. In case of hardware failure which occurs due to natural calamities, it's not possible to fail, occurs due to faults in programs that are also handled by the recovery manager.

The replication manager stores data at different servers in different parts and the data is encrypted. There is a backup i.e. data warehouse in the whole data, whenever the data

at different servers is lost due to an attack or any faults. The particular part is fetched from the data warehouse. In case, if the whole server crashes then the data warehouse acts as primary data. Processing is managed by Fault Prediction Manager and Fault Masking Manager. Fault Prediction Manager works as RAM which maps the memory, processor of virtual machines and whenever there is a new resource that needs to be stored in these virtual machines, the manager first checks the RAM and allocates the machine according to the space it has. Fault Masking Manager works as a time checker, which is present in every virtual machine and checks the execution time of processors, in this if the execution time of the processor exceeds it will lead to failure. Therefore, before sending a request to any machine at the time the checker will first check its execution time and if the execution time is more the request will be discarded as it can lead to failure (Rawat, Sushil, Agarwal, Sikander, et al., 2021).

Fault tolerance techniques enable systems to perform tasks in the presence of faults. Fault tolerance can be achieved through some kind of redundancy. The most common method used is checkpoint-restart; an application is restarted from an earlier checkpoint or recovery point after a fault. This may result in the loss of some processing and applications may not be able to meet strict timing targets. This is considering hybrid architecture in terms of implementing the Checkpointing is primarily used to avoid losing all the useful processing done before a fault has occurred. Checkpointing consists of intermittently saving the state of a program in a reliable storage medium. Upon detection of a fault, the previous consistent state is restored. In case of a fault, checkpointing enables the execution of a program to be resumed from a previous consistent state rather than resuming the execution from the beginning. In this way, the amount of useful processing lost because of the fault is significantly reduced, proactive and reactive policies, reactive policies including replication and resubmission methods, and proactive, including monitoring (Rehman et al., 2022).

Symbol	Meaning
VMListfault	List of failure virtual
hostfr	Failure rate of a host
hostListfault	List of hosts containing failure virtual machines
thresfr	Threshold value of remaining execution time
thresret	Threshold value of failure rate
hostListmigration	List of hosts whose virtual machine needs to be migrated
hostListother	List of hosts whose virtual machine needs to be either replicate or retry
hostListretry	List of hosts whose virtual machine needs to retry
hostret	Remaining execution time of host that is an aggregation of all the *faultyVMret* in the respective host
hostListrep	List of hosts whose virtual machine needs to replicate
faultyVMret	Remaining execution time of faulty virtual machine

Table 1: Notations

This is considering hybrid architecture in terms of implementing the proactive and reactive policies, reactive policies including two algorithms from each approach. This proposed architecture is divided into three phases: pre-processing phase, execution phase, and resubmission phase. When the client sends a task to execution, the pre-processing phase begins to calculate a replica heuristic based on the algorithm that is used to calculate a different replica for each task. Then the information has been sent to the scheduler to allocate each replica for its VM. In addition, store any updated information in the schedule buffer.

Many calculations have to be performed during the execution phase such as reliability assessment (rel), execution time, expected execution time, and task status. After an execution request if all replicas or at least one of the same tasks has passed status the client received a success signal. Then the execution phase requests the next task from the schedule buffer to perform. However, if all replicas of the same task have failed status the execution phase sends a fault message to the fault tolerance algorithm (FTA) (resubmission technique). The FTA chooses the VM which has a large value of reliability assessment to submit the failed task for re-running. If a task fails, again the resubmission phase begins to resubmit the task again to the same VM. If tasks fail again

resubmit it again until not exceed the max resubmission count (res_count). If we exceed the max resubmission count, the client receives a rejected message (meaning this fault may be in the task).

The objective of the proposed approach is to monitor and anticipate faulty VMs. When there is an occurrence of a faulty VM, the proposed approach will identify the spare for the VMs running on the failing host based on the threshold values.

Failure Predictor Module (FPM)

A failure predictor is a process that can predict the fault, which is going to take place in a VM at a certain point in time. It periodically estimates the list of nodes that are expected to fail. In this paper, FPM based on the stochastic model is used to predict VM failure. The working of the FPM is depicted in the figure below.

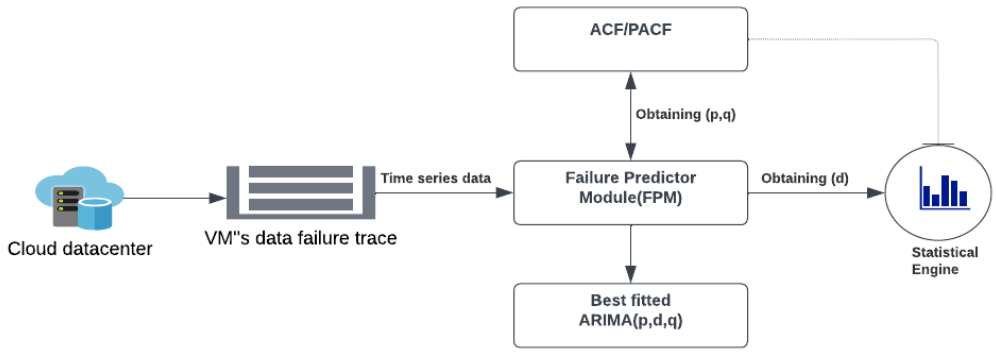

Figure 4. 1: Proposed TBAHFT Approach

In this particular study, the impact of fault predictor on FT techniques is calculated. With the use of FPM, a better FT technique that can reduce the failure rate in the cloud is proposed. In the TBAHFT approach, there are two different methods, FPM and utilization of VMs to predict the fault. An ARIMA (Auto Regressive Integrated Moving Average) process is a generalization of an ARMA (Auto Regressive Moving Average) process. Generally, the ARIMA is denoted by *(p, r, q)* where $p \geq 0$, $r \geq 0$, $q \geq 0$ and p, r, $q \in Z$, p is the order of autoregressive model, r is the degree of differencing, and q is the order of the moving average model. For a time series of data Xt, ARMA *(p, r)* model is given by: (1)

$$-\sum_{i=0}^{p'}(\beta iLi)Xt \; = \varepsilon t \sum_{i=0}^{q} \gamma i L^{i}$$

Equation 1: ARMA (p, r)

where t ∈ Z, $X_t \in R$, $\beta_o = -1$, $\gamma_0 = 1$, L is the lag operator, ε_t are error terms, γ_i are the parameters of the moving average part and β_i are the parameters of the autoregressive part of the model.

Now assume that $\sum_{i=0}^{p'}(\beta iL)\, i$ satisfies $\sum_{i=1}^{p'}(\beta iLi)$

and

$$\int \frac{dk}{dLk}\ \left[\sum_{i=0}^{p'}(\beta iLi)\right] = 0,\ 0 \leq k \leq \text{r-1 at L=1}$$

$$\frac{\int dr}{\int dLk}\ \left[\sum_{i=0}^{p'}(\beta iLi)\right] = cont, at\ L = 1 \text{ then it can be written as;} \quad (2)$$

Equation 2: ARIMA const at L = 1

$$\sum_{i=0}^{p'}(\beta i\, L^i) = \left(1 - \sum_{i=1}^{p} \phi iLi\right)(1-L)^d \quad (3)$$

Equation 3: ARIMA at φi Li

which can be further simplified to

$$\left(1 - \sum_{i=1}^{p} \phi iLi\right)Y = \left(1 + \sum_{i=1}^{q} \gamma iLi\right)\varepsilon t\ where\ \text{Yt} = (1-L)^{\text{r}}Xt \quad (4)$$

Equation 4: ARIMA when Yt = (1 − L)rXt.

Now, the forecast can be made for the process Y_t, using the general method of autoregressive forecasting.

TBAHFT Model

The TBAHFT requires the existence of a failure predictor that can periodically estimate the expected VM failures in the system. The proposed approach uses FPM, which implements a stochastic model to predict the VM failures in the virtual cluster. This paper uniformly described the failure prediction as categorical, where the FPM forecast whether a failure event will occur or not in the near future. The list of faulty VMs is predicted by the FPM that are injected to TBAHFT to implement the adaptive technique before the failure occurs in the system. The proposed approach's objective is to ensure that all the failure-prone VMs in the virtual cluster should have a healthy node associated with them at any given point in time. The overall working of the approach is illustrated in Figure 2. It takes different fault tolerant actions during runtime at decision-making points. The different possible actions that can be taken as migrate, replicate, retry, and no-action.

The TBAHFT receives the input from the FPM to get the status of each VMs running in the virtual cluster. In response to the failure prediction by FPM, the proposed approach initiates the process to decide the best-ft action based on the input.

Let us define the following sets.

H_f =set of failed hosts.

$H_{s(i)}$=set of sparse hosts.

Let the failure probability of each host in H_f be $P(H_{f(i)})$, i = 1, 2, 3 . . . .n, and V_i=set of VM's in $H_{f(i)}$.

Let $V_{f(i)j}$ be the set of VM's failed in host$H_{f(i)}$ then $V_f(i)_j \subseteq V_i$, j = 1, 2, 3, . . . k_i (where k_i = total number of failures in$H_{f(i)}$).

Suppose P_{thres} - $P((H_{f(i)}) = \alpha_i \in Q$ (where Q is the set of rationals) where P_{thres} denotes the threshold probability.

Define an indicator function

$$\mu Q- = \begin{Bmatrix} = -1 \\ 0 \end{Bmatrix}$$

Equation 5:Indicator function Q-

if otherwise $\alpha i \in Q-$ (5)

where Q^- is the set of negative rationals.

Merge $V_{fm}(i)j \subseteq V_{f(i)j}$ into $H_{s(i)}$ for which $\mu Q^- = 1$ where $V_{fm}(i)j$ denotes the set of failure VM to be merged. Let $H_{o(i)} \subseteq H_{f(i)}$ be the set of hosts for which $\mu Q^- = 0$ and $V_{fo(i)j}$ be the set of VM's failed in host $H_{o(i)}$ then $V_{fo(i)j} \subseteq V_{f(i)j}$.

Suppose P_{th_ret} - $P(V_{f(i)j}) = \beta_{ij} \in Q$ (where Q is a set of rationals and P_{th_ret} denotes the remaining execution time threshold probability).

Define an indicator function

$$\mu Q_- = \begin{cases} 1 \ if \ \beta ij \ \in \ Q \\ 0 \ otherwise \end{cases} \quad (6)$$

Equation 6:Indicator Function execution time thr

Restart $V_{fo(i)j}$ for which $\mu Q^- = 1$ and Replicate $V_{fo(i)j}$ for which $\mu Q^- = 0$.

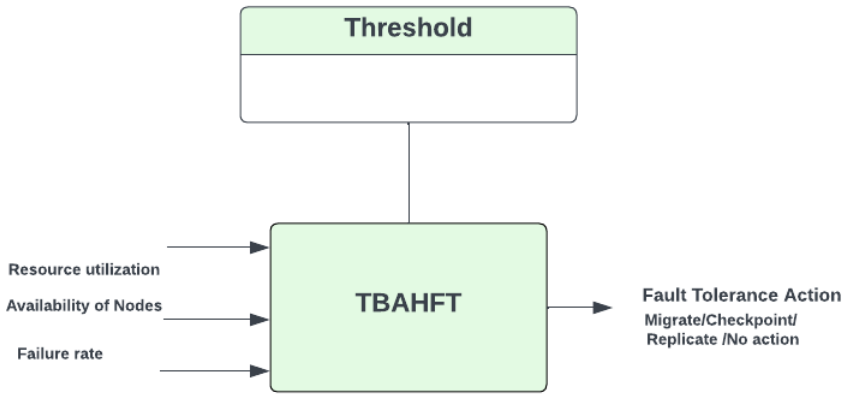

Figure 4. 2: TBAHFT model

A threshold-based mechanism is implemented to guarantee cloud service availability and reliability in TBAHFT. The objective of the proposed approach is to monitor and anticipate faulty VMs. When there is an occurrence of a faulty VM, the proposed approach will identify the spare for the VMs running on the failing host based on the threshold values. The major component of the TBAHFT architecture is AM, responsible for determining the most suitable FT action during VM failure as shown in Figure above. Designing an optimized AM is challenging (Cheraghlou et al., 2019).

First, it must consider cloud application performance factors. Second, it includes the available spare nodes, costs, and benefits of different FT techniques. It consists of three significant components: classifier, migrator, and replicator.

(1) The classifier receives a VMListFault as input from FPM. It first identifies all hosts that contain that VMs from VMListfault and generates the hostListfault list. After that, it calculates the hostfr from hostListfault based on the equation (7).

$$hostfr = no\ of\ failure\ VMs\ / \ total\ no\ of\ VM$$

Equation 7:Hostfr

(2) The major task of Migrator is to identify an additional spare node from the virtual cluster to relinquish an unhealthy node. It compares the hostfr with the thresfr of all the hosts from hostListfault and creates two lists hostListmigration and hostListother. The hostListmigration comprises hosts having a failure rate greater than thfr and hostListother contains all the remaining hosts from hostListfault. After that, it sorts all

the hosts from hostListfault based on hostfr in descending order. Finally, it migrates all the VMs from the hostListmigration

(3) Replicator compares the remaining execution time of a faulty VMs from the hostListother with thresret.

It replicates all the faulty VMs within the same host with more remaining execution time than thresret and retry all the remaining VMs from the hostListother. The replication and retry are done based on the maximum remaining execution time.

Suppose different hosts are denoted as {H1,H2, . . .Hn} and spare nodes {S1, S2, . . . Sm} in the virtual cluster. Spare nodes can be reserved either during application submission or allocated at runtime by the resource manager.

The AM chooses a preventive action based on the evaluation of the failure rate of each faulty host. It can take four possible fault tolerance actions against the faulty VMs.

(1) Migration: When the hostfr is greater than thresfr. All the VMs are migrated to another healthy spare node to avoid imminent failure.

(2) Replication: When the remaining execution time of the application in the failure VM is greater than thresret. All the failed VMs are replicated on the same host based on increasing the remaining execution time.

(3) Retry: When the remaining execution time of the application in the failure VM is less than thresret. All the failed VMs are restarted based on increasing the remaining execution time.

(4) No Action: When the impact of fault is trivial or the virtual cluster is in a fault-free state. It avoids the unnecessary overhead and doesn't take any action.

To calculate the reliability of a cloud system, you will need to know the throughput (how many requests can be processed in a given time), failure rate (how many requests result in errors), and migration time (the amount of time it takes for the system to move data between machines).

The reliability of the cloud system is determined by calculating the mean time between failures (MTBF) - this is calculated by taking the total throughput, subtracting any failures and dividing this number by the total number of requests made.

$$Reliability = \frac{Total\ No\ of\ throughput - total\ failures}{total\ no\ of\ requests}$$

Equation 8: Reliability

Another way to calculate reliability is to divide throughput by failure rate. This is known as MTTR or mean time to recover. It measures how quickly the system recovers from an error or outage. First, calculate throughput and failure rate separately then the reliability.

$$Throghput = \frac{No\ of\ requests}{time}$$

Equation 9: Throughput

$$Failure\ rate = \frac{No\ of\ failures}{no\ of\ requests}$$

Equation 10: Failure rate

After calculating the throughput and failure rate we pass into calculating

$$Reliabilty = \frac{Throughput}{Failure\ rate}$$

Equation 11: Reliability

Reliability of a cloud system can be calculated by comparing its execution time and migration time. It is important to note that the reliability of a cloud system should be measured in two parts: resource utilization and task execution times.

Resource utilization is determined by looking at the number of resources available and how these are used. For instance, if there are eight cores available could eight tasks be processed simultaneously? If the answer to this question is yes then the reliable of the system is higher than if only six cores were used for all requests.

The second part of determining reliability is looking at task execution time. This measures how long it takes for a request to be completed from when it was sent until it has finished executing its goal. Factors such as periods of massive usage or varying levels of congestion can affect this metric significantly.

Finally, migration time can also help determine the reliability of a cloud system since it measures how long it takes for workloads to move from different locations (like from one cloud provider to another). This can be determined by calculating the time needed for different operations performed within a network such as establishing connections, command propagation and synchronization, etc.

By comparing these three metrics (resource utilization, task execution times and migration time), we can accurately calculate the reliability of a given cloud system, migration time and execution time can be used as an indicator of reliability. If a system is able to migrate large amounts of data quickly and effectively, then it likely has high levels of reliability. However, if migration times are high, then it indicates that there may be problems with the service that could lead to downtime or other issues.

$$Reliabilty = \frac{Total\ Migration\ time}{failure\ rate}$$

Equation 12: Reliability using Migration

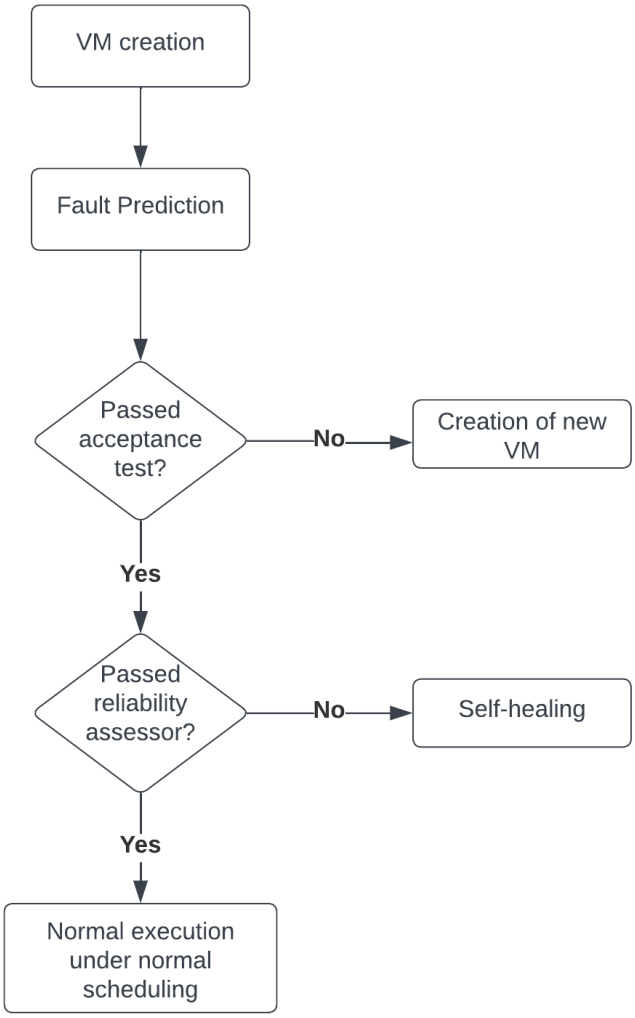

Figure 4. 3: Flow graph of hybrid fault tolerance approach

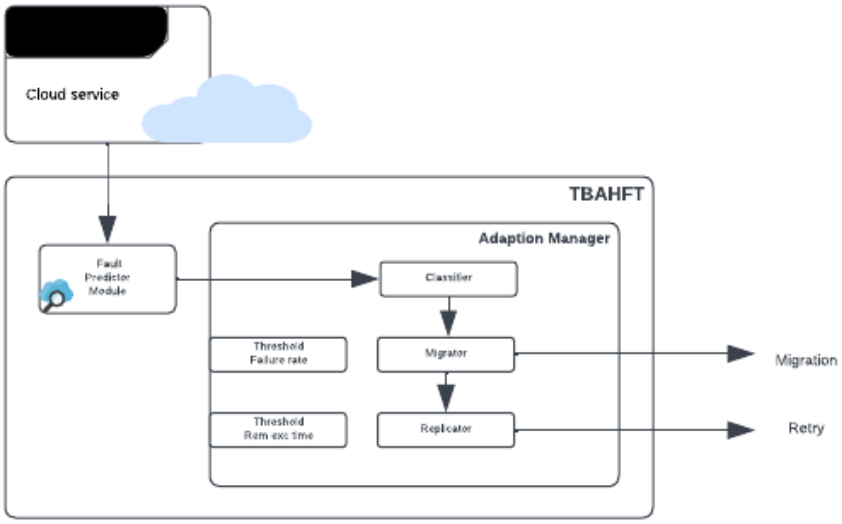

Figure 4. 4: TBAHFT Architecture

Algorithm 1 Threshold based adaptive fault tolerance algorithm (TBAHFT)

1: **Input:** hostList, VmListfault

2: **Output:** hostListfault

3: //Create a list of faulty host from all the hosts that contains the faulty VMs

4: **procedure** (createFaultyHostList (hostList, VmListfault))

5: **for** each host in hostList do

6: **if** any VM from VmListfault exist in host **then**

7: hostListfault ← host

8: **else**

9: No action

10: **return** hostListfault

1: **Input**: host

2: **Output**: hostfr

3: //Calculate the failure rate of the faulty hosts

4: **procedure** (failureRate(host))

5: hostfr ← (host.no_of _failure_VMs / host.no_of _running_VMs) * 100

1: **Input**: hostListfault, thresfr

2: **Output**: hostListmigration,hostListother

3: //Classify the faulty hosts into two categories - migration and others

4: **procedure** (Classifer(hostListfault, thresfr))

```
5:    for each host in hostListfault do
6:        if failureRate(host) ≥ thresfr then
7:             hostListmigration ← host
8:        else
9:            hostListother ← host
10:        return (hostListmigration, hostListother)
```

1: **Input:** hostList, thresfr, thresret

2: **Output**: hostListmigration, hostListother

3: //Threshold based adaptive fault tolerant approach

4: **procedure** (TBAHFT(hostList, VMListfault, thresfr, thresret))

5: hostListfault ←createFaultyHostList(hostList,VMListfault)

```
6:    (hostListmigration,hostListother)←classifer(hostListfault,thresfr,thresret)
7:    sort(hostListmigration)
8:    for each host in hostListmigration do
9:        for each VM in host do
10:           VM migration to spare host Si
11:   sort(hostListother)
12:   for each host in hostListother do
13:       for each faultyVM in host do
14:           if faultyVMret ≥ thresret then
15:               replicate faultyVM
16:           else
17:               retry faultyVM
```

The pseudocode for the TBAHFT technique is presented in Algorithm 1. The classifier, failure Rate and create FaultyHostList procedures are invoked in the TBAHFT process. The createFaultyHostList procedure identifies the faulty hosts in which failure VMs are present. The host failure rate hostfr is calculated by the failureRate procedure. The classifier procedure categorizes the faulty hosts into two different lists hostListmigrate for migration and other hostListother based on the threshold value of the failure rate. TBAHFT sorts the hostListmigrate and migrates the VMs to the spare hosts in a distributed manner. The distributed approach implements the optimized allocation based on a power-aware policy for distributing workloads across available hosts. Also, the VMs from hostListmigrate migrate to underutilized hosts and efficiently manage the overall energy across the host. It also differentiates the VMs from hostListother to replicate or retry based on a thresret.

4.1 Workflow

The algorithmic workflow of the proposed approach for cloud application is presented in Figure 7. The work procedure of TBAHFT describes as follows:

(1) FPM predicts the list of faulty VMs using the stochastic model in the cloud application.

After faulty VMs are predicted list of faulty hosts are going to created in accordance to the proposed algorithm. Moreover, the workflow of the faulty VMs prediction is described below.

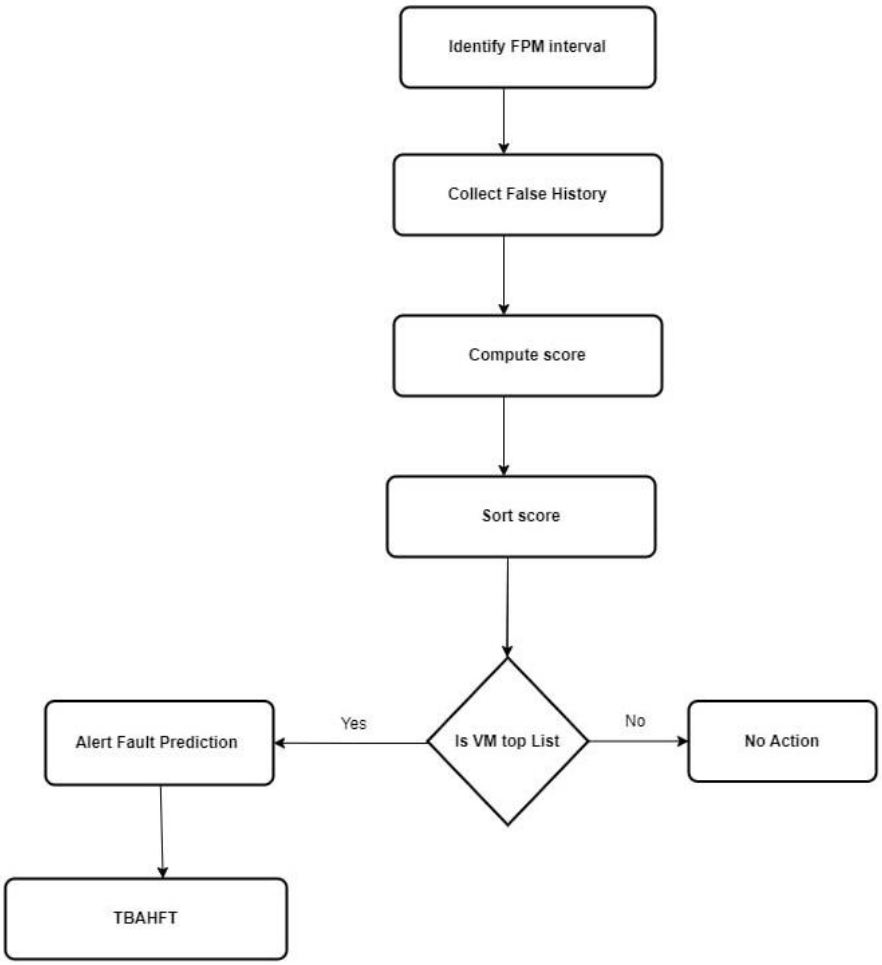

Figure 4. 5: Faulty VMs Predictions

1. Identify and define the time interval in which the FPM is expected to perform the prediction.

2. Collect information regarding fault history such as system usage, faults detected by monitoring agents and logs for each VM.

3. Compute a probabilistic score for each VM based on the collected information using a stochastic model.

4. Compare the computed scores and sort them from highest to lowest probability of fault occurrence in descending order to create a list of probable faulty VMs.

5. Generate an alert for each VM on the list based on their probabilities of experiencing a fault to notify stakeholders about potential issues before they occur. In this specific work, the impact of fault predictors on FT techniques is calculated. With the use of FPM, a better FT technique that can reduce the failure rate in the cloud is proposed. In the TBAF T approach, there are two different methods, FPM and utilization of VMs to predict the fault.

(2) AM obtains the list of faulty VMs from step1 to reduce the effect of VM failure. It then creates a hostListFaulty that contains the faulty VMs.

(3) AM calculates the hostfr of all the faulty hosts based on equation (7).

(4) Classifier generates two host lists from hostListfault namely hostListmigration and hostListother based on a thresfr. The hostListmigration contains all the hosts having a failure rate greater than thresfr and hostListother contain the remaining hosts from hostListfaulty.

(5) Migrator (a) Sort the hostListfaulty based on hostfr in descending order.

(b) Migrate the VMs to different spare hosts in a distributed fashion.

(6) Replicator generates two host lists from hostListother based on a threshold value of remaining execution time thresret. The hostListrep contains all the hosts having a remaining execution time greater than thresret and hostListretry contains the remaining hosts from hostListother.

(a) Sort both hostListrep and hostListretry based on hostret in descending order.

(b) Replicate the VMs from the hostListrep within the same host.

(c) Retry the VMs from hostListretry within the same host.

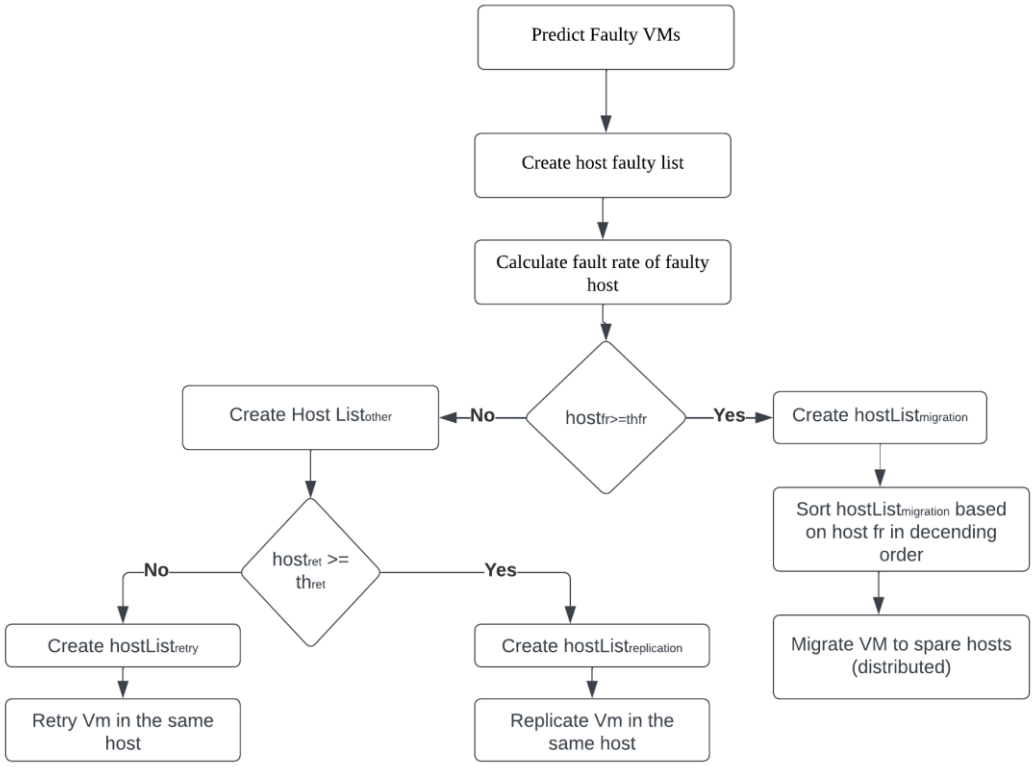

Figure 4. 6: TBAHFT Workflow

Reliability is ensured through fault tolerance which in turn includes two techniques namely proactive and reactive fault tolerance. By including these techniques into the Cloud system, failure is either prevented by pre-techniques or will never be efficient in the occurrence of a fault.

A Cloud system consisting of purely proactive or reactive techniques will never be efficient in the occurrence of a fault. The proactive techniques will prevent the occurrence of the fault but it doesn't provide any solution to recover from the damage if a fault occurs. A reactive system provides ways to minimize damage and recovery methods but does not provide any preventive mechanisms. Considering these factors it is clear that only a system consisting of both proactive and reactive techniques edition or damage control is done after the failure occurs.

To avoid VM failure due to memory insufficiency or variable capacity change dynamic load balancing in the VM can be done before applications start to run on it. To ensure higher reliability fault tolerance should be provided during the runtime also i.e. when

applications have started to run on the VM. The scheme proposed here provides fault-tolerant techniques when the applications have started to run on the VM.

Auto Regressive Integrated Moving Average (ARIMA) is a popular statistical technique used in time series forecasting. It is a type of regression that combines both autoregressive and moving average components into a single model. ARIMA can be used to predict the future values of a variable by looking at the past values of the same variable, as well as other related variables. The use of ARIMA models has become particularly relevant in cloud computing due to their effectiveness in predicting workload patterns and VM migrations. As VM migration is an expensive operation, efficient prediction models such as ARIMA are very useful for maximizing resource utilization and thus cost savings for users.

First Fit: This type of host selection policy is used in virtual machine (VM) environments to find a suitable physical server on which to run a VM instance. The First Fit algorithm iterates through all the available servers, testing them for whether they meet the capacity and resource requirements specified for the VM. Once it identifies a server that meets those requirements, the VM can be deployed on it.

Least Full: This type of host selection policy evaluates all available physical servers and chooses the server with the least resources currently being used for VMs. It attempts to spread out existing load across more servers, helping to balance load across the entire system and maintain efficiency.

4.2 Experimental Setup

This section evaluates the effectiveness and efficiency of TBAHFT through a simulation experiment using CloudSim Simulator. In the experiment, three different use cases are considered. In the first case, the virtual cluster size was set to 15, the number of VMs used was 50, and 500 parallel applications were running. Secondly, the virtual cluster size was 30, and the 100 VMs and 1000 parallel applications were running. In the last case, a 60 size virtual cluster was used where 200 VMs and 2000 parallel applications were running. The threshold value of thresfr and hostret is set to 50 %. In all the above cases, the TBAHFT is compared with the other two FT techniques, No Fault Tolerance (NoFT) and NDTBAFT (Non-Distributed Threshold Based Adaptive Fault Tolerance). The distributed approach implements the optimized

allocation based on a power-aware approach for distributing workloads across available hosts.

In this approach, the VMs from hostListmigration migrate to underutilized hosts and efficiently manage the overall energy across the host. The Non distributed approach doesn't consider any optimized allocation approach, which is deterministic. In all three FT techniques, the two variants of host selection approach, namely first fit and least full, are implemented. A brief overview of all the techniques is as follows:

(1) NoFT First Fit: No fault tolerance strategy is used and the host selection policy is First Fit.

(2) NoFT Least Full: No fault tolerance strategy is used and the host selection policy is Least Full.

(3) NDTBAFT First Fit: A variant of TBAHFT that does not apply a distribution approach during migration and uses the First Fit policy as the host selection policy.

(4) NDTBAFT Least Full: A variant of TBAHFT that does not apply a distributed approach during migration and uses Least Full as the host selection policy.

(5) TBAHFT First Fit: Apply distributed approach during migration and use the First Fit policy as the host selection policy.

(6) TBAHFT Least Full: Use a distributed approach during migration and use a least full host selection policy.

All of the above methods are evaluated by using the four performance metrics:

(1) Total execution time: The total time taken by the virtual cluster to jointly execute a set of parallel applications including the time consumed in executing runtime and system services.

(2) Total Migration time: The total time spent from when the migration of VMs is initiated from the source hosts until the VMs are resumed on the destination hosts.

(3) Failure rate: The ratio of the total number of failed applications to the total number of applications submitted to a host.

(4) Throughput: The ratio of the total number of finished applications executed to the total number of applications submitted.

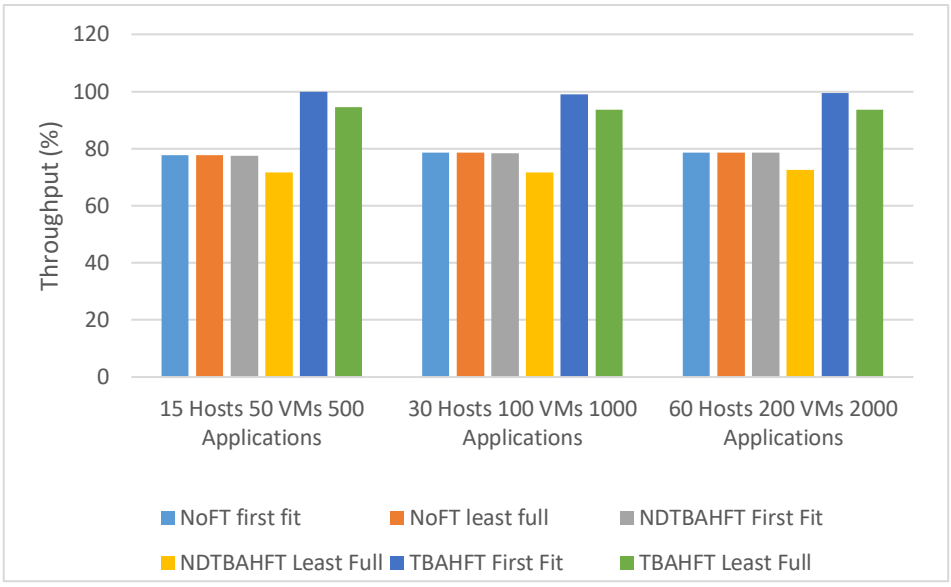

Figure 4. 7: Throughput

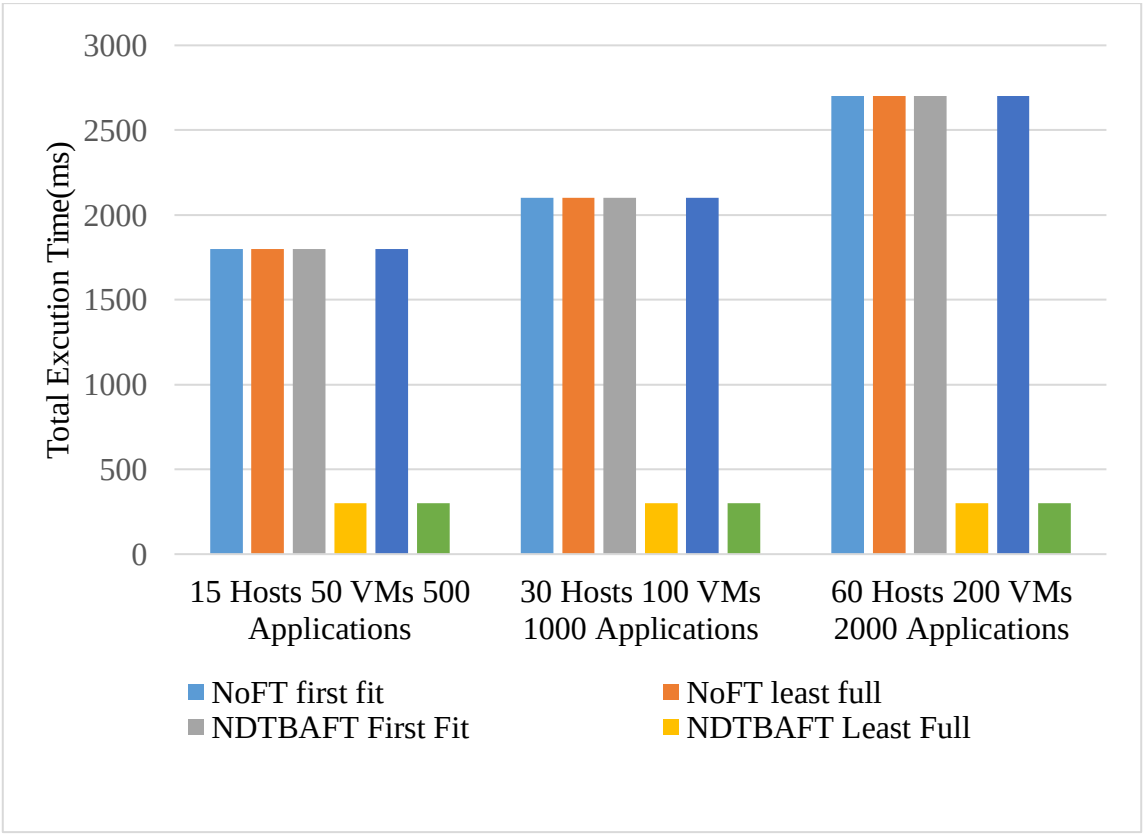

Figure 4.8: Execution Time

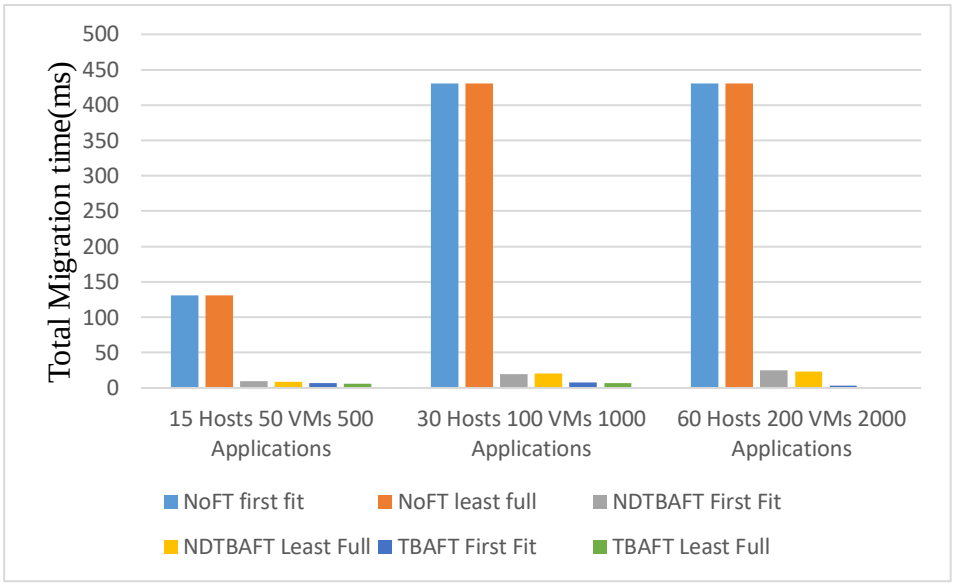

Figure 4.9: Total Migration

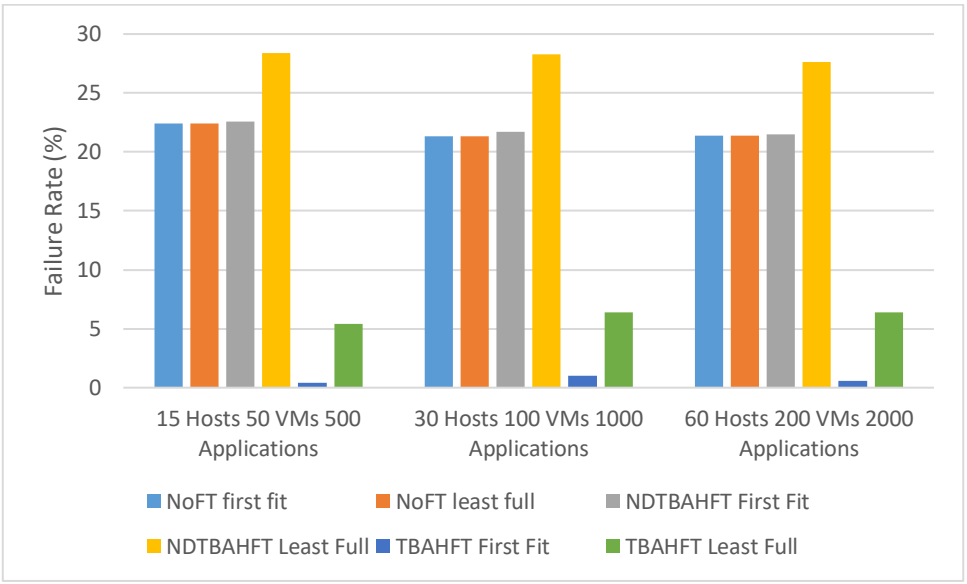

Figure 4.10: Failure Rate

Comparison of used allocated and requested hosts

	15 Hosts 50 VMs 500 Applications Hosts			**30 Hosts 100 VMs 1000 Applications Hosts**			**60 Hosts 2000 VMs 2000 Applications Hosts**		
FT Techniques	**Used**	**Allocated**	**Requested**	**Used**	**Allocated**	**Requested**	**Used**	**Allocated**	**Requested**
NoFT First Fit	1	12	15	1	25	30	1	50	60
NoFT Least Full	7	12	15	19	25	30	49	50	60
NDTBAFT First Fit	3	12	15	3	25	30	7	50	60
NDTBAFT Least Full	4	12	15	3	25	30	8	50	60
TBAHFT First Fit	6	9	15	14	18	30	28	35	60
TBAHFT Least Full	6	9	15	14	18	30	20	20	60

Table 2: Comparison of used allocated and requested hosts

Comparison of used allocated and requested hosts

	15 Hosts 50 VMs 500 Applications Hosts			30 Hosts 100 VMs 1000 Applications Hosts			60 Hosts 2000 VMs 2000 Applications Hosts		
12									
FT Techniques	Migration	Replication	Retry	Migration	Replication	Retry	Migration	Replication	Retry
NoFT First Fit	46	0	0	96	0	0	196	0	0
NoFT Least Full	46	0	0	98	0	0	196	0	0
NDTBAFT First Fit	32	0	2	64	0	2	82	0	6
NDTBAFT Least Full	28	0	2	66	0	2	78	0	7
TBAHFT First Fit	23	3	2	22	20	4	8	24	30
TBAHFT Least Full	18	3	2	19	22	4	0	5	50

Table 3: Comparison of migration replication and retry

CHAPTER FIVE

5. Conclusion and Future work

5.1 Conclusion

Threshold based adaptive fault tolerance has shown to be a superior algorithm compared to its non-distributed, no fault tolerance first fit counterparts. This is confirmed by the metrics of reliability, availability and execution time which all demonstrate that the algorithm of threshold based adaptive hybrid fault tolerance produces better results in terms of system performance than other algorithms. By comparing these algorithms threshold based adaptive hybrid fault tolerance outperforms in both reliability, availability and execution time when confronted with failure or outage events. Its advantages ensure that it will remain the preferred algorithm for utilization as cloud computing technology continues to expand in the world.

The average throughput percentage compared to the TBAHFT is 97.1 compared to 77.6 of NoFT First Fit, 77.6 of NoFT Least Full, 77.4 of NDTBAFT First Fit and 71.6 NDTBAFT Least Full when 15 hosts with 50 VMs and 500 parallel applications are running. Similarly, the throughput when 30 hosts with 100 VMs and 1000 parallel applications, 60 hosts with 200 VMs and 2000 parallel applications in different scenarios showed the same result. Thus, the proposed algorithm outperformed the previous algorithms.

The total execution time of Threshold Based Adaptive Hybrid Fault Tolerance (TBAHFT) is be better than the rest of the algorithms by the execution of task per milliseconds in different scenarios of hosts, VMs and applications running in parallel manner. Thus, the total execution time of the proposed algorithm has shown a better performance. Based on the above metrics we can deduce that the availability and reliability of any given cloud system or application is directly proportional to the highest throughput, fastest execution and migration time in case of failure. So, TBAHFT is the superior algorithm when considering reliability, availability and execution time metrics. Compared to other algorithms such as Non Distributed Threshold Based Hybrid Fault Tolerance and No Fault Tolerance First Fit, TBAHFT offers the highest rate of reliability, the most availability and the shortest execution time. All this combined leads to TBAHFT being the clear choice when considering fault tolerance in modern computing ecosystems. TBAHFT provides users with reliable fault tolerance easily making it an invaluable asset to any organization seeking a dependable

experience with their technology assets. Its superior performance on all three front makes it one of the best algorithms available in terms of fault tolerance today.

5.2. Contribution

TBAHFT is a major contributor when combining the two FT approaches of threshold based adaptive fault tolerance and minimal reconfiguration along with masking, the system is experimented to have an optimized migration time, reduced failure rate, shorten execution time and increased throughput due to performance monitoring and intelligent adaptation compared to other algorithms. Thus, the reliability and availability of the cloud system are high. The TBAHFT allows system components to dynamicall adapt to changing conditions and workload variations while the minimal reconfiguration technique minimizes the resource overhead needed for post-fault recovery. As a result, application running time is significantly reduced, failure rates are minimized, and overall system performance is increased.

5.3. Future work

Even though optimized migration time, reduced failure rate, shorten execution time and increased throughput cost reduction metrics has not been considered, and only the static migration of VMs were considered in this particular study. Thus, we would like to encourage the future researchers to address the problems in the future.

Special Acknowledgment

This research project is funded by Adama Science and Technology University under the grant number:

ASTU/SM-R/502/22
Adama, Ethiopia

References

Abdelfattah, E., Elkawkagy, M., & El-Sisi, A. (n.d.). *Hybrid Fault Tolerance Approach Based on Replication Impact Heuristic for Cloud Computing A Reactive Fault Tolerance Approach for Cloud Computing View project Intrusion detection systems View project Hybrid Fault Tolerance Approach Based on Replication Impact Heuristic for Cloud Computing*. https://www.researchgate.net/publication/338867287

Alqahtani, S. M., & Arishi, H. (n.d.). A REVIEW ON THE TOOLS AND TECHNIQUES FOR EFFECTIVE FAILURE DETECTION AND PREDICTION IN CLOUD COMPUTING. *Management, & Applied Sciences & Technologies*, *11*(16), 11–16. https://doi.org/10.14456/ITJEMAST.2020.315

Amin, Z., Sethi, N., & Singh, H. (2015). Review on Fault Tolerance Techniques in Cloud Computing. In *International Journal of Computer Applications* (Vol. 116, Issue 18).

Amoon, M. (2015). *A Framework for Providing a Hybrid Fault Tolerance in Cloud Computing*. www.conference.thesai.org

Amoon, M. (2016). Adaptive Framework for Reliable Cloud Computing Environment. *IEEE Access*, *4*(c), 9469–9478. https://doi.org/10.1109/ACCESS.2016.2623633

Buyya, R., Ranjan, R., & Calheiros, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *Proceedings of the 2009 International Conference on High Performance Computing and Simulation, HPCS 2009*, 1–11. https://doi.org/10.1109/HPCSIM.2009.5192685

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599–616. https://doi.org/10.1016/j.future.2008.12.001

Cheraghlou, M. N., Khadem-Zadeh, A., & Haghparast, M. (2019). A new hybrid fault tolerance approach for internet of things. *Electronics (Switzerland)*, *8*(5). https://doi.org/10.3390/electronics8050518

Dhinesh Babu, L. D., & Venkata Krishna, P. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft*

Computing Journal, *13*(5), 2292–2303. https://doi.org/10.1016/j.asoc.2013.01.025

Elshaer, I. A. (2012). Munich Personal RePEc Archive What is the Meaning of Quality? *MPRA Paper*, *57345*. https://mpra.ub.uni-muenchen.de/57345/1/MPRA_paper_57345.pdf

Ganesh, A., Sandhya, M., & Shankar, S. (2014). A study on fault tolerance methods in Cloud Computing. *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, 844–849. https://doi.org/10.1109/IAdCC.2014.6779432

Ganga, K., & Karthik, S. (2013). A fault tolerent approach in scientific workflow systems based on cloud computing. *Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, PRIME 2013*, 387–390. https://doi.org/10.1109/ICPRIME.2013.6496507

Gibson, J., Rondeau, R., Eveleigh, D., & Tan, Q. (2012). Benefits and challenges of three cloud computing service models. *Proceedings of the 2012 4th International Conference on Computational Aspects of Social Networks, CASoN 2012*, 198–205. https://doi.org/10.1109/CASoN.2012.6412402

Goundar, S., Bhardwaj, A., & Studies, E. (2018). *Efficient Fault Tolerance on Cloud Environments*. *8*(3), 20–31. https://doi.org/10.4018/IJCAC.2018070102

Hasan, M., & Goraya, M. S. (2018). Fault tolerance in cloud computing environment: A systematic survey. In *Computers in Industry* (Vol. 99, pp. 156–172). Elsevier B.V. https://doi.org/10.1016/j.compind.2018.03.027

INFORAMTIONQ.com. (n.d.). *Cloud Computing –Benefits, Services and Deployment Models*.

Kastouni, M. Z., & Ait Lahcen, A. (2022). Big data analytics in telecommunications: Governance, architecture and use cases. In *Journal of King Saud University - Computer and Information Sciences* (Vol. 34, Issue 6, pp. 2758–2770). King Saud bin Abdulaziz University. https://doi.org/10.1016/j.jksuci.2020.11.024

Khan, S. (2019). Cloud Computing: Issues and risks of Embracing the Cloud in a Business Environment. *International Journal of Education and Management Engineering*, *9*(4), 44–56. https://doi.org/10.5815/ijeme.2019.04.05

Kumari, P., & Kaur, P. (2021). A survey of fault tolerance in cloud computing. In *Journal of King Saud University - Computer and Information Sciences* (Vol. 33,

Issue 10, pp. 1159–1176). King Saud bin Abdulaziz University. https://doi.org/10.1016/j.jksuci.2018.09.021

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., & Leaf, D. (n.d.). *NIST Cloud Computing Reference Architecture Recommendations of the National Institute of Standards and Technology*.

Lutkevich, B. (n.d.). *Quality of Service (QoS)*. Retrieved December 13, 2021, from https://www.techtarget.com/searchunifiedcommunications/definition/QoS-Quality-of-Service

Mell, P. M., & Grance, T. (2011). *The NIST definition of cloud computing*. https://doi.org/10.6028/NIST.SP.800-145

Mesbahi, M. R., Rahmani, A. M., & Hosseinzadeh, M. (2018). Reliability and high availability in cloud computing environments: a reference roadmap. *Human-Centric Computing and Information Sciences*, *8*(1). https://doi.org/10.1186/s13673-018-0143-8

Mohammad, O. K. J. (2018). Recent trends of cloud computing applications and services in medical, educational, financial, library and agricultural disciplines. *ACM International Conference Proceeding Series*, 132–141. https://doi.org/10.1145/3233347.3233388

Nazari Cheraghlou, M., Khadem-Zadeh, A., & Haghparast, M. (2016). A survey of fault tolerance architecture in cloud computing. In *Journal of Network and Computer Applications* (Vol. 61, pp. 81–92). Academic Press. https://doi.org/10.1016/j.jnca.2015.10.004

Osorio, G. A., del Real, C. S., Valdez, C. A. F., Miranda, M. C., & Garay, A. H. (2006). Effect of inclusion of cactus pear cladodes in diets for growing-finishing lambs in central Mexico. *Acta Horticulturae*, *728*, 269–274.

Rawat, A., Sushil, R., Agarwal, A., & Sikander, A. (2021). A New Approach for VM Failure Prediction using Stochastic Model in Cloud. *IETE Journal of Research*, *67*(2), 165–172. https://doi.org/10.1080/03772063.2018.1537814

Rawat, A., Sushil, R., Agarwal, A., Sikander, A., & Bhadoria, R. S. (2021). A New Adaptive Fault Tolerant Framework in the Cloud. *IETE Journal of Research*. https://doi.org/10.1080/03772063.2021.1907231

Rehman, A. U., Aguiar, R. L., & Barraca, J. P. (2022). Fault-Tolerance in the Scope of Cloud Computing. *IEEE Access*, *10*, 63422–63441. https://doi.org/10.1109/ACCESS.2022.3182211

R, K. G. (2015). Fault Tolerance in Cloud Using Reactive and Proactive Techniques. *International Journal of Computer Science and Engineering Communications*, *3*, 1159–1164. www.scientistlink.org

Savu, L. (2011). Cloud computing: Deployment models, delivery models, risks and research challanges. *2011 International Conference on Computer and Management, CAMAN 2011*. https://doi.org/10.1109/CAMAN.2011.5778816

Sivagami, V. M., & Easwarakumar, K. S. (2015). Survey on Fault Tolerance Techniques in Cloud Computing Environment. *International Journal of Scientific Engineering and Applied Science (IJSEAS)*, *1*. www.ijseas.com

Sun, D., Chang, G., Miao, C., & Wang, X. (2013). Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments. *Journal of Supercomputing*, *66*(1), 193–228. https://doi.org/10.1007/s11227-013-0898-7

Vuyyuru, M., Annapurna, P., Babu, K. G., & Ratnam, A. S. K. (2012). *An Overview Of Cloud Computing Technology*. http://blogs.idc.com/ie/?p=190

Zhang, P., Shu, S., & Zhou, M. (2018). An online fault detection model and strategies based on SVM-grid in clouds. *IEEE/CAA Journal of Automatica Sinica*, *5*(2), 445–456. https://doi.org/10.1109/JAS.2017.7510817

Zhou, A., Wang, S., Cheng, B., Zheng, Z., Yang, F., Chang, R. N., Lyu, M. R., & Buyya, R. (2017). Cloud service reliability enhancement via virtual machine placement optimization. *IEEE Transactions on Services Computing*, *10*(6), 902–913. https://doi.org/10.1109/TSC.2016.2519898

Cheung WKW; Lam HKH; Chiu STC; Ke LYF(2018). “A Hierarchical Fully-Meshed Interconnect Network Topology towards Higher Fault Tolerance for High Performance Clusters” IEEE Transactions on Parallel Distributed Systems,, 30(2), pp. 337 - 350

Couromulles N.; Fish M.; Janjic B.; Guffanti A.(2018). “Testing Software Fault Tolerance Is Worth Doing” IEEE Computer Graphics & Applications vol 38(4), pp 90 - 98

Jalali S.; Bateni SM.; Lavasani MM.(2019). “Real-time Fault Detection Algorithm for Autonomous Unmanned Vehicles” 8th International Symposium on Computational Intelligence & Robotics,.pp 363 – 368

Li JG.; Wu J.; Li YJ.; Li C.(2017). “Improving Load Balance under Elastictiy Functional Will Improve Reliability for Cloud Computing Utilization” Proceedings 14th International Conference on Computers& Education vol 37(2),pp 310 – 318

Sample Codes

Java code to create 30 hosts with 100 VMs and 1000 parallel applications on CloudSim

```
public class CreateVM {

 public static void main(String args) {

  int numHost = 30; // number of hosts needed

  int numVMs = 100; // Number of VMs required

  int numApps = 1000; // Parallel Applications

              /Creating a cloudsim object for simulating /

         CloudSim simulationEnvironment = new CloudSim();

          /Start the simulation /

        simulationEnvironment.init();

        / Creating the list of Hosts/

        List<Host> hostList=new ArrayList<Host>();

        / Creating the host and adding them to the list /

         for(int i=0 ; i<numHost ; i++){

           Host single_host = new Host("HOST" + i); // Initializing Host without any
parameters

           hostList.add(single_host);                                    }

     /Adding all of the VMs to VMlist /

     List<Vm> vmList=new ArrayList<Vm>();

     / Creating VMs and adding them to vmlist /

        for(int j=0 ; j<numVMS ; j++){

            Vm singlevm = new Vm("VM" + j); //Initializing VM without any parameters
like configuration

             vmList.add(singlevm);                    }

      /Adding all applications to Applist /

      List <App> appList = new ArrayList();
```

```
/ Creating Application and Adding them in applist/

for(int k=0 ; k<numApp ; k++){

App single_application = new App("APP" +k); // Initializing APPs without anyParameters like range,type etc .

appList .add (single_application); }

/Setting Up executing time for Applications/ simulationEnvironment . setTime (100); } }public static void main(String args) { // Starting point of the program

Log.printLine("Starting..."); // Logging message for starting the program

try { // Exception Handling Used so that even if error arises in the middle it wont stop execution and move on to rest code

int numuser = 15; // Setting number of users as 15

Calendar calendar = Calendar.getInstance(); // Creating a Calendar Instance for handling event list in the simulation

boolean traceflag = false; // Adding Trace flag for easier debugging when printed on standard output

CloudSim.init(numuser, calendar, traceflag);

@SuppressWarnings("unused")

Datacenter datacenter0 = createDatacenter("Datacenter0"); /// Creating a new Datacenter with name 'Datacenter0'

Broker broker = createBroker(); //Creating a Broker Object to handle incoming cloudlets into datacenters

int brokerId = broker.getId(); //Storing ID at which Broker is initialised

createHosts(15); // Creating 15 Hosts using Helper Method Created outside main class definition

vmsToDatacenter(brokerId); // Passing our newly created broker id through helper method to assign each VM to Datacenters

///Creating 500 applications with random priorities and assigning each application to VMs which will be handled by createApplications() method defined outside main class definition

for (int applCount=1 ; applCount<=500 ; applCount++){
```

Application app =WhatcreateApplication(); Count.submitApplication(app); /// Submitting each application generated from helper method } } catch (Exception e) { e.printStackTrace(); } finally { Log.printLine("Finished TAAF Algorithm Simulation"); }} private static Datacenter createDatacenter(String name) /this public method is used to initiate cloud simulation based on Algorithm using threshold technique/ { List<Host> hostList = new ArrayList<Host>(); ///Instantiate a list object (hostlist)//// List<Pe> peList1 = new ArrayList<Pe> (6);

Creating 60 Hosts, 200VMs and 1000 applications

public class ThresholdBasedAHFT {

MAXHOSTS = 60; MAXVM = 200 ; MAXAPP= 2000 ;

private static List<Cloudlet> cloudletList;

//Create an array to store virtual machines(vms)

private static List<Vm> vmList;

public static double[][] hostvsnoofappmatrix= new double[MAXHOSTS]MAX_APP; // stores the matrix of no:of apps on each host with host id as indexing key

public static double appvsAvaiNoopMatrix= new doubleMAX_APP2; //stores the matrix for avaiable no of operations for each application with appid as indexing key

public static DatacenterBroker broker , tempbroker :

public static void main(String args) {

initialize(); //initialise the environment

performAssignment(); /perform assignment amongst created hosts, VMs and applications</code> </pre> pre><code>static void assignApplications(){ for (i = 0 ; i <=MAXHOSTS;++i{</code></pre><pre><code>or(j=0 ; j <= MAXAPP){if(hostvsNoofappmatrixij!=1){assignApplicationToHost(i,j);}}}static assignapplicationtohost(int host, int App){ hostvsnoofappmatrixhostApp=1; int NoToOps = appsofOperations(App); if((NoToOps + thresholdvalue >= (availabelOperationPerHost) { acoAlg = new acoalgorithm() acoAlg./setIndexAtHostsWhereApplicationDoenstExist(host); acoAlg./setAppwithThresNNewOpersChanges(); acoAlg ./ completeAssignedWork(); return true/sussessful or false/unsucessful in Assigning Applications } else {return false;} } </code></pre>

```
      int numuser = 1;   // number of grid users
       Calendar calendar = Calendar.getInstance();
       boolean traceflag = false
       CloudSim.init(numuser, calendar, traceflag);
    int totalVMs=1000;
       int VMAttempts=2;
      double timeThreshold=100000
  cloudList = new ArrayList<Cloud>();
      datacenterList = new ArrayList<Datacenter>();

      for (int i=0 ; i < 5000 ; i++){

        int responseTime=i/4 + 1000;
         for (int j=0 ; j < 100 ; j++) {
           if(responseTime>timeThreshold){


startMyAftAlgorithm(VMAttempts,totalVMs}          }                              }
}          }                    private static void startMyAftAlgorithm(int VMAttempts,
int totalVms){                  for (Cloud c : cloudList) {                  for (Datacenter
d : datacenterList) {                          LoadBalancingBalancer balancer = new
LoadBalancer(c,d);
balancer.rebalanceVMsandHosts(VMAttempts,totalVms);                  }       } }
```