Ternate Fault Tolerance in Cloud Environment

Anwar Ahamed Shaikh[1], Dr Shish Ahmad[2]
[1]Department Of Computer Science and engineering,
Integral University
Kursi Road, Lucknow, Uttar Pradesh 226026
[2]Department Of Computer Science and engineering,
Integral University
Kursi Road, Lucknow, Uttar Pradesh 226026

Abstract: Cloud computing is a method of storing and managing data on remote servers over the internet, and then accessing that data via the internet. In cloud computing, fault tolerance is creating a plan for continuing current work even if a few components are unavailable. Many existing works have been done to overcome the fault in cloud virtual machines. Even though it produces low throughput, loss of virtual machine interaction, minimized network path, and low availability of cloud virtual machines. In this research, the technical complexities in achieving fault tolerance are removed by using Ternate fault Tolerance in cloud environment, in which the virtual machine fault with low throughput is eliminated by using Compeer wield which provides interaction between the virtual node and increases throughput. Moreover, to maximize the network path, Escalate lane is proposed to store all information about the process and to eliminate network failure. Finally, the availability of virtual machines is increased by eternal Alacrity. Thus the Ternate Fault Tolerance in Cloud Environment can be outperforming the other existing techniques with higher throughput.

Keywords: Fault Tolerance, Baseline in the virtual machine, Photostat, Throughput, updating virtual machine, Reserved Memory

1. INTRODUCTION:

Cloud computing is an all-encompassing approach for providing IT as a service [1]. Cloud computing is a type of computing in which services are delivered via the internet utilizing models and abstraction layers [2]. Cloud computing is a distributed computing paradigm that was created to provide dynamic computing services through the internet. It enables users to remotely access, configure, and alter resources (such as software and hardware) [3]. "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) [4] that can be quickly provisioned and released with the least management effort or service provider interaction," according to the US National Institute of Standards and Technology (NIST).

Cloud computing consists of three levels of service models such as [5] (a) Software as a Service (SaaS): cloud service providers provide software applications to consumers or end-users as services. (b) Platform as a Service (PaaS): cloud-based platforms for developing, running, testing, and managing applications. (c) Infrastructure as a Service (IaaS): cloud-based access to physical machines, storage, networks, servers, and virtual machines.

Cloud computing's flexibility comes from the ability to allocate resources on demand [6]. Many research concerns, like as fault tolerance and [7] security, are fully handled in the cloud. Fault tolerance is a critical issue in the cloud; it refers to all of the approaches that enable a

system to accept software faults [8] that remain after it has been developed. Failure recovery, lower costs, and increased performance are the major advantages of adopting fault tolerance in cloud computing [9]. There is a fault when many instances of an application are operating on various virtual machines and one of the servers goes down, fault tolerance is used at this issue.

When using cloud infrastructure for real-time applications, the possibilities of mistakes rise. Due to the distance between the cloud nodes (virtual computers) and the transceiver (job submitting node, actuator, or sensor). Because many real-time systems are also safety-critical, they need a higher level of fault tolerance. [10] Cloud resources shorten the time to market and lower the cost of execution, but they are also vulnerable to resource breakdowns. Hardware failures, software failures, and virtualization, among other things, are the most common causes of failure. Task faults, virtual machine (VM) faults, and scientific workflow level faults are the three types of errors that occur at the point of a scientific workflow. To avoid failure, which may result in financial loss as well as injuries, safety essential real-time systems must function effectively. As a result, there is a greater requirement to accept failure when such systems are utilized with cloud infrastructure.

To cope with service dependability concerns, several methods have been created that either anticipate and avoid failures or attempt to offer fault tolerance. [11] In a complex computer environment like a cloud computing system, where VM failures are unavoidable, detecting and removing errors that may develop in the system is impossible. As a result, fault tolerance techniques are increasingly being utilized [12] to allow the system to service the request even if some of the components are not functioning properly. A fault-tolerant system should be able to function despite software or hardware [13] component failures, power outages, or other types of unanticipated adversity.

In existing cloud methods, several fault tolerance schemes such as checkpointing and replication have been utilized. The use of [14] VM replication to ensure the stability of applications running on VMs hosted on cloud hosts has been advocated. In cloud computing, faults and breakdowns of physical equipment are unavoidable. Each physical computer can contain many virtual machines, each of which can be running a different job or application. If a virtual machine fails, the job is halted, and any work that has already been completed is lost. Similarly, when a physical machine (PM) fails, all VMs hosted by that physical machine fail as well. [15] Each virtual machine must be replicated to achieve fault tolerance. If a VM or the host machine fails, it is mapped to a replica VM, and the completed tasks are carried out at the replica, providing users with service dependability.

Fault tolerance enables systems to provide required services despite component failures or one or more defects. Fault tolerance techniques assist in identifying and addressing system faults that may arise as a result of hardware or software failures. Fault tolerance is especially important in cloud platforms because it gives consumers confidence in the performance, dependability, and availability of cloud-based applications. Fault tolerance methods for cloud-based systems are often reactive, aiming to mitigate the impact of failure after it has happened. This research seeks to provide solutions to various issues like virtual machine fault, network fault, and availability fault in a cloud environment. As a result, Ternate Fault Tolerance in Cloud Environment has been proposed to address these issues. The main contributions of this paper are as follows:

- Compeer Wield technique is used to remove the virtual machine fault by providing virtual machine interaction.
- Escalate Lane technique is used to tolerate the network fault by creating a ruler node for information storage.

- Eternal Alacrity technique uses reserved memory to update the data and thereby removes the availability fault.

Thus the proposed Ternate Fault Tolerance in Cloud Environment solves the issues like virtual machine fault, network fault, and availability fault in a cloud environment.

The content of the paper is organized as follows: section 1 represents the introduction; section 2 presents the literature survey of fault tolerance; the novel solutions are presented in section 3; the implementation results and its comparison are provided in section 4; finally, section 5 concludes the paper.

2. LITERATURE SURVEY

Zhou et al [16], proposed a checkpoint-based fault tolerance method. An optimum checkpoint technique that is aware of edge switch failures. Two methods are used in the edge switch failure-aware checkpoint approach. For checkpoint image storage server selection, the first approach uses the data center structure and connection characteristics. The second method selects the recovery server based on the checkpoint image storage feature as well as the data center structure. The root layer, the aggregation layer, and the edge layer are the three levels that make up a fat-tree data center network. "In the same subnet" refers to the host servers that share the same edge switch. "In the same pod" refers to the host servers that share the same aggregation switches. The edge switches link the host servers to the network. The simulation findings demonstrate that the checkpoint-based approach in cloud computing systems may effectively assure service dependability while reducing root layer network resource consumption. Taking checkpoints regularly and restarting a failed service from the latest stored checkpoint image in the network path, on the other hand, is time-consuming and has significant overheads during normal operations.

Chen et al [17], proposed a replication-based fault tolerance method. By enclosing the protected programs inside the virtual machine and regularly checkpointing the whole virtual machine (VM) state to the backup replication, virtualization provides a straightforward means of ensuring high availability. According to the study, existing VM replication solutions suffer from high checkpointing costs, network delay, and needless CPU resource usage in backup replication. The COLO++ system, which is based on COLO, is a non-stop service solution for client-to-server systems with coarse-grained lock-stepping VMs. To increase the dependability and minimize access latency, an asymmetric virtual machine replication approach is used, but high availability of fault tolerance in a cloud environment could not be declared.

Rezaeipanah et al [18], proposed a fuzzy-based fault tolerance method. Cloud computing is quickly becoming one of the most useful new technologies. Failures must be foreseen and handled to minimize their influence on the system and guarantee proper job execution. Only a few recently developed fault tolerance approaches have concentrated on the non-fault detection dimension. This study examines the nature of the mistake and how it is detected in depth. It also includes a fuzzy-based approach for preparing an acceptable error tolerance answer. Requesting a task re-execution and migration procedures through the checkpoint are used to enhance error tolerance and load balancing when an error occurs. One of the primary considerations for guaranteeing service availability and dependability, as well as job performance, is fault tolerance. A new fuzzy-based technique for diagnosing and responding to fault management has been presented. Faults may be recognized properly using the suggested architecture without the need for a precise input dataset.

Khaldi et al [19], proposed scientific workflow in cloud computing. Complex scientific procedures need a lot of computing power. To carry out such complicated operations, cloud computing resources are a viable option. Task clustering is a useful approach for reducing system overhead and improving the fine computational granularity of jobs in a scientific process that runs on distributed resources. Despite their considerable influence on large-scale distributed resources, such as Clouds, previous clustering algorithms overlook the effect of failures on the system. In this study, researchers introduce FT-HCC, a new fault-tolerant task clustering approach that is aimed to improve workflow performance by adding workflow execution time and execution cost restrictions. The researchers present a novel method for fault-tolerant clustered job scheduling. When compared to a previous approach utilized in WFMS and planned on a Cloud distributed resource environment, the algorithm dramatically improves workflow and execution cost. Implementation with different nodes was not optimized in this approach.

Ray et al [20], proposed a proactive fault tolerance technique. Cloud federation is a novel computing paradigm that allows cloud service providers (CSPs) to share their spare resources (virtual machines) with other CSPs when demand for such resources is low. The computing environment of federation members must be fault-tolerant to ensure the dependability and availability of cloud services provided through the federation. In this work, we present a proactive fault tolerance system based on CPU temperature that prevents failures inside the federation. In the case of a problem, we suggest using an algorithm called Preference-Based Fault Management (PBFM) to manage the federation. ILP-based model for VM redistribution in a federated cloud environment in the event of failures. While redistributing VMs, this ILP solves the multi-objective optimization issue of maximizing profit while minimizing migration costs. The defective physical machine of CSPs inside the federation is predicted using a proactive fault tolerance technique based on CPU temperature. To address the problem of an arbitrary number of defective CSPs at a given moment, an algorithm (PBFM) based on dominance and preference relationships has been suggested.

Effective fault tolerance framework could not be detected [16] restart service fails due to network path in checkpoint [17] an asymmetric virtual machine replication approach could not give high availability of fault tolerance in the cloud environment. [18] re-execution and migration results in ineffective load balancing in nodes [19] adding execution time and cost improves workflow does not provide fault tolerance in the cloud environment. In [20] Proactive fault tolerance could not suggest different relationships since the preference of nodes has been used. By considering all these kinds of faults in virtual machines, networks, and availability a novel framework should be developed so that it could be implemented in the cloud environment. The proposed methods will contribute to fault tolerance in the cloud, with high throughput, and increase the availability of virtual machines. The approaches that are already in use in fault tolerance are explained above. The next section explains the techniques and benefits of the proposed method.

3. TERNATE FAULT TOLERANCE:

Cloud computing systems improve the service dependability based on replication and checkpoint techniques, virtual machines in the cloud increase the efficiency of the services. A huge number of services and applications can run via the internet using virtual machines in the form of nodes. If a failure happens during the execution of an application or computing, it will cost more in terms of time, money, and power as well as harm cloud service providers reputation. Many existing works have been done to overcome the fault in cloud virtual machines. Even though it

produces low throughput, loss of virtual machine interaction, minimized network path, and low availability of cloud virtual machines. To overcome these issues a novel framework needs to be proposed to overcome virtual machine fault, network fault, and availability fault in the cloud. A novel framework Ternate Fault Tolerance is innovated to overcome the above-mentioned issues. While connecting numerous VM in a network, virtual machines could not be controlled and cause virtual machine faults. So a novel technique Compeer Wield is proposed in which virtual machine kinship is done based on Photostat. So the fault in virtual machines is interacted and controlled that increase the throughput. Even though throughput has been increased and tolerates the virtual machine faults, network fault occurs due to failure in the destination node. To maximize the network path a novel, Escalate Lane is proposed in which a ruler node is created that stores all information about the process and thereby maximizes the network path. After maximizing the network path there is a need to increase the availability of virtual machines for efficient processes in the cloud environment. To increase the availability a novel Eternal Alacrity is proposed, in which the updating virtual machine is processed in reserved memory without pausing any virtual machines. As a result, the proposed framework effectively tolerates the fault in the cloud environment.

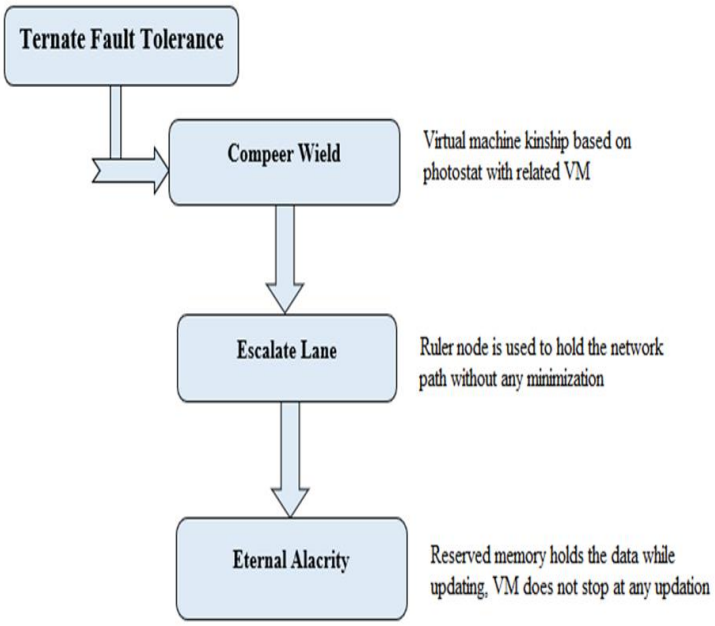

Figure 1: Block diagram of the proposed method

3.1 Compeer Wield:

Fault tolerance is a major concern for ensuring the availability and dependability of vital services in the cloud environment. Failures to minimize the impact on the system and application performance should be optimized and proacted. To predict the problems and take necessary

action before they occur, fault tolerance techniques are utilized. Many existing works have been done to overcome the fault in cloud virtual machines. Although it results in low throughput, loss of virtual machine interaction, a reduced network path, and low cloud virtual machine availability. When more virtual machines are connected in a network, it is difficult to control the virtual mechines and hence virtual machine fault occurs. To overcome the fault in virtual machines, the Compeer Wield technique is proposed. In Compeer Wield, the virtual mechine relationship is established based on Photostat. Photostat offers a technique to make the design and analysis that use shared registers much easier. Shared registers are extended by Photostat-based virtual machines. Photostat allows a method to build consistent global states of shared storage without interfering with system execution and deal with several photostat operations at once rather than one at a time. The Photostat feature is most useful when it is required to preserve the state of the virtual mechine. so it can return to the same state repeatedly. When a photostat is taken, the data on all of the virtual machine's disks are saved including the data related to whether the virtual machine was switched on, shut off, or suspended. Virtual machine photostats are only supported by virtual machine fault tolerance when the fault-tolerant virtual machine can be backed up using storage photostat to protect against data corruption or storage failure.

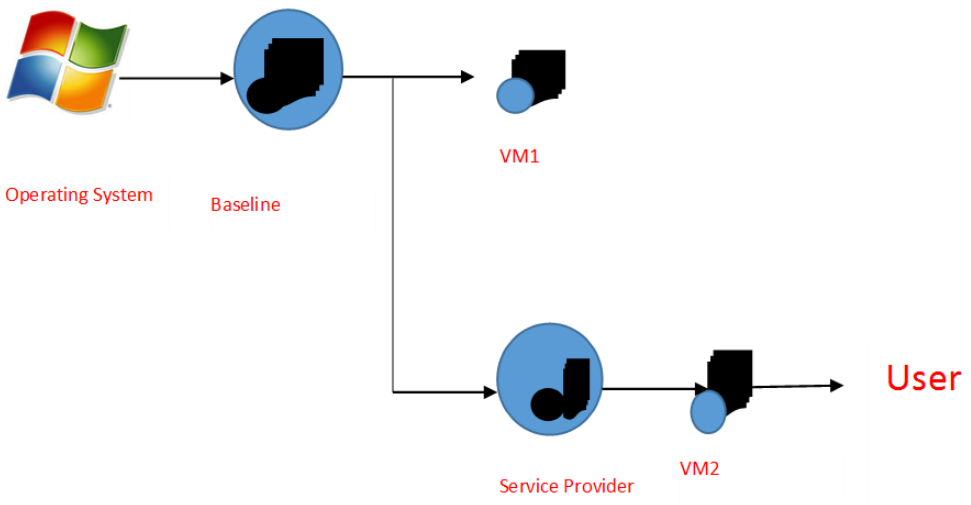

Figure 2: Photostat to preserve every VM

The steps involved in using templates to backup FT virtual machines to set up the virtual computer are as follows

- Clone a template of the virtual machine before turning on FT for the virtual machine.
- For the virtual machine, turn on FT.
- For the FT virtual machine, enable the in-guest backup program.

Storage Photostatting differs from VM snapshots in that it is a capability offered by the backend storage array.

The steps involved in using storage Photostat to back up FT virtual machines are as follows

- Photostat the virtual machine files using storage Photostatting.
- Create a new virtual machine and register it.
- Transfer the vm file to another host, but disable FT for this virtual machine.

- Back up the freshly registered non-fault tolerant virtual machine with VM Consolidated Backup.

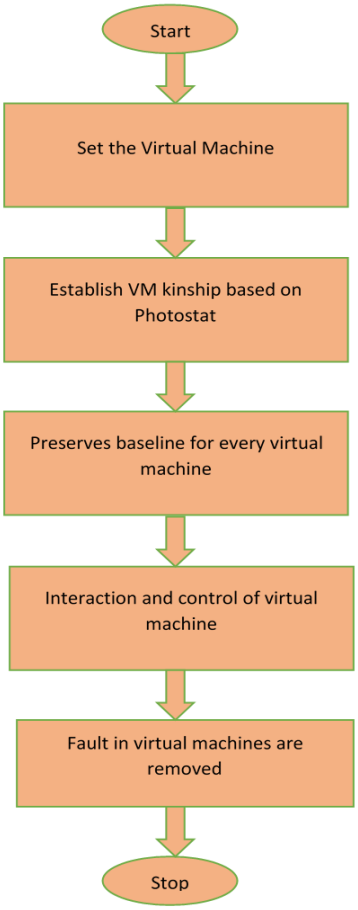

Figure 3: Flowchart for Compeer Wield Technique

Photostat saves a virtual machine's state and data at the moment the photostat is taken. When a photostat is taken from a virtual computer, it is copying and stores a picture of the virtual machine in a certain state. When it is needed to revert to a virtual machine state frequently but don't want to build numerous virtual machines, photostats come in handy. By collecting multiple photostats of a virtual computer, it is simple to create repair sites in a linear procedure. To accommodate a range of work methods, keep various locations with a large number of photostats. Individual virtual computers are used for photostats. Taking a photostat of numerous virtual machines, such as a virtual computer for each team member, necessitates taking a separate photostat of each team member's virtual machine. Photostats are important for testing software with unknown or potentially detrimental consequences in the short term. A photostat can be used as a restore point during a linear or iterative process, such as installing update packages, or during a branching process, such as installing various versions of software. The use

of photostats ensures that each installation begins at the same point. Photostat preserves the baseline for every virtual machine and the photostat achieves the process according to every related virtual machine.

The following information of the virtual machine is preserved by photostat:

- Configure the virtual machine: The virtual machine directory, which contains the disks that were added or modified after the photostat was taken.
- The condition of power: The virtual computer can be started, stopped, or paused at any time.
- The condition of the disk: The current state of all virtual drives in the virtual machine.
- The condition of memory: The memory contents of the virtual machine.

Since the fault in virtual machines is interacted and controlled by using the Compeer Wield technique and hence the throughput is increased by using the compeer wield technique. Even though throughput has been increased and tolerates the virtual machine faults, network fault occurs due to failure in the destination node. The next section explains the next technique, Escalate Lane.

3.2 Escalate Lane:

The Compeer Wield technique removes the virtual machine failure and thus increases throughput but the network failure occurs due to an insufficient network path. Escalate Lane technique is used to remove the destination node failure by enhancing the network path. In order to eliminate the network failure caused by inadequate network path, Escalate Lane technique creates a ruler node to store all the information about the process. The following steps are required to create Ruler Node:

- Use FsImage, the file system metadata replica, to construct a new Ruler Node.
- Data nodes must be configured to recognize the formation of this new Ruler Node.
- After it has completely loaded the final FsImage checkpoint, Ruler Node will resume its operation, and the cluster will return to normality.

The Ruler Node works with tens of thousands of data nodes to fulfill requests from client applications. FsImage is a file in the Ruler Node's local file system that contains the whole namespace of the file system. All of the file system's directories and file inodes are serialized as well. Each inode is an internal representation of a file or directory's information. The ruler node is now built, and it contains all of the process information. If any failure occurs in the destination node, the ruler node is activated immediately in the place of failure and it holds the network path without any minimization and thereby removes the network failure and enhances the network path.

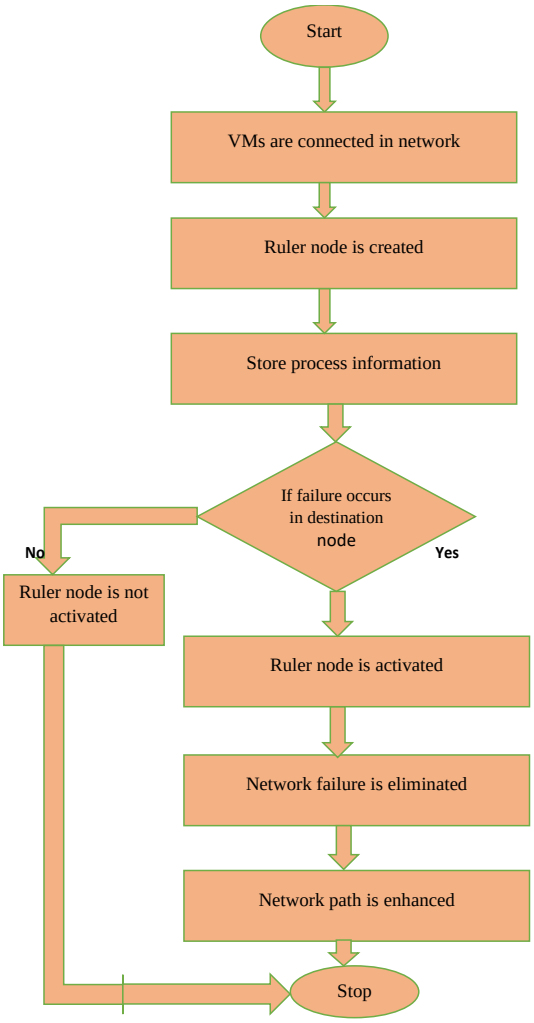

Figure 4: Flowchart for Escalate Lane

After maximizing the network path there is a need to increase the availability of virtual machines for efficient processes in the cloud environment. Existing techniques are available exclusively on the virtual machines processed and other virtual machines remain in the idle mode which leads to lesser availability of virtual machines. The next section explains the next technique, Eternal Alacrity.

3.3 Eternal Alacrity:

The Escalate Lane technique removes the network failure and thus increases the network path but the availability of virtual machines is not sufficient. Eternal Alacrity technique is used to increase the availability of virtual machines and at the same time, the Eternal Alacrity technique makes the virtual machine in active mode by using reserved memory. The term "reserved memory" refers to storage space set aside for a technology's usage. The notion is that memory set aside for one process can't be accessed by other processes. While traditional computers had a specific amount of reserved memory for their core processes and other amounts of memory reserved for programs, virtual machines in more sophisticated network virtualization systems may have different kinds of memory reservations, some of which can be changed by

programmers. The idea of memory reservation can be applied differently to these newer and more complex systems since network fertilization includes the construction of virtual data storage spaces rather than physical PCs or workstations. Each virtual machine consumes memory in accordance with its allocated size, as well as virtualization overhead RAM. The amount of RAM accessible to the guest operating system is determined by the size chosen. This is distinct from the virtual machine's real RAM allotment. The amount of memory demand, as well as the host's resource parameters (sharing, reservation, and limit), determine the memory usage. FT's major goal in cloud computing is to provide high availability in order to fulfill client criteria for service performance and completion time as specified by the service-level agreement. Because an FT service is an integral element of the service-level objective, having one in a cloud environment is critical. For this cloud computing system FT method is employed using reserved memory. This approach uses the Eternal Alacrity technique, which updates VM based on reserved memory, to provide a high-availability system in the event of failures. For updating the virtual machine, there is a need to upgrade the guest operating system or an application that's running on it and a need to change either the virtual machine itself or the policies that apply to it or the need to add a new virtual machine to the package. If this virtual machine isn't working properly, it can be removed from the stored memory list. According to the concept, requests from the host machine's or physical server's resources generate a collection of virtual machines without pausing. The VM monitor, which is either software, hardware, or firmware that generates and operates virtual machines, does this. The host machine is the server on which the virtual machine manager (VMM) executes guest virtual machines. The VMM provides virtual operating platforms to guest operating systems and controls the execution of these guest operating systems. The VMM keeps track of all virtual nodes produced by various host servers. It also keeps and preserves records from reserved memory during the process of assigning a task to a virtual node of a certain host server.

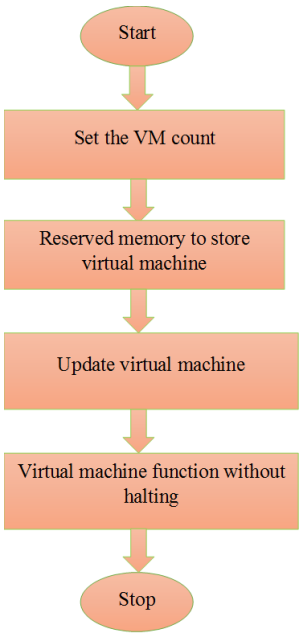

Figure 5: Flowchart for Eternal Alacrity

So the availability terms are always high since the virtual machine does not pause in any situation, such that all the virtual machines can be reacted without any fault. Overall the Ternate Fault Tolerance in Cloud includes three major techniques. The first is the Compeer Wield technique which increases the throughput and removes the virtual machine fault. Second, the Escalate Lane technique increases the network path and thereby eliminates network failure. Finally, the Eternal Alacrity technique is used to increase the availability of the virtual machine and thereby removes the idle state of the virtual machine. Thus Ternate Fault Tolerance in Cloud Environment provides fault tolerance to cloud computing efficiently. The next section explains the results obtain from the Ternate Fault Tolerance in Cloud Environment and discusses it in detail.

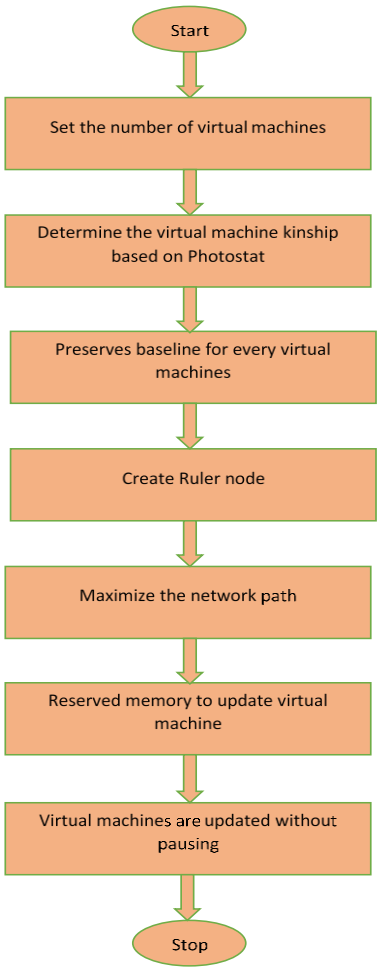

Figure 5: Overall Flowchart of the Proposed system

4. RESULT AND DISCUSSION:

This segment provides a detailed description of the implementation results as well as the performance of the proposed system and a comparison section to ensure that the proposed system performs valuable.

4.1 Experimental Setup:

This work has been implemented in the working platform of java with the following system specification and the simulation results are discussed below.

Platform **:** Java
OS **:** Windows 7
Processor **:** 64-bit Intel processor
RAM **:** 8 GB RAM

4.2 Performance metrics of the proposed method:

4.2.1 Throughput:

The number of tasks in a workflow that has finished their execution per unit time is referred to as throughput. A system's throughput, or the number of jobs that complete execution in a given amount of time, should be high.

Throughput can be calculated using the following formula

$$T = \frac{I}{F} \tag{1}$$

Where,

I - the number of units in the production process (Inventory)

F - The time the inventory units spend in production from start to finish

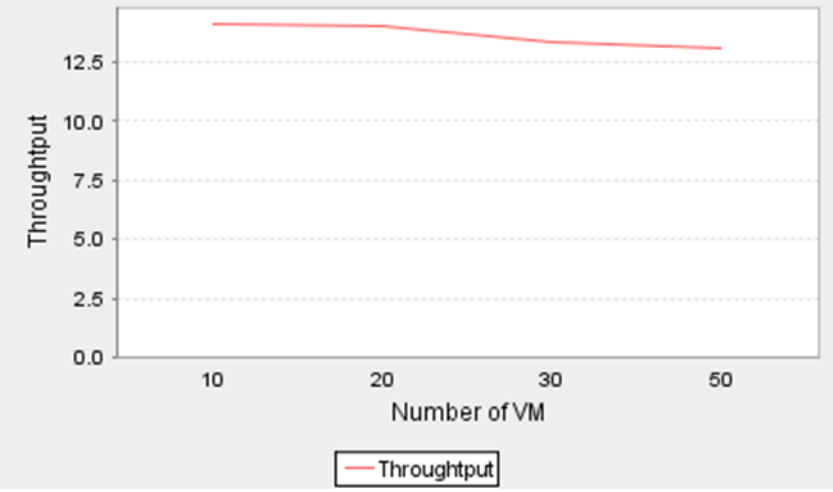

Figure 6: Overall throughput of the proposed system

The above-mentioned graph clearly explains the throughput of the proposed system. The throughput of the proposed system attains the maximum value of 14 and the number of virtual machines is taken from 1 to 50. The above graph depicts that as the throughput increases the number of VM also increases. Initially, the increase is large but as the number of VM attains a limit the throughput starts to decrease gradually. The throughput of the proposed system is increased by using the Compeer Wield technique in which the fault in a virtual machine is removed.

4.2.2 End to end delay:

The time it takes for a packet to go from source to destination across a network is referred to as end-to-end delay or one-way delay (OWD). It's a popular word in IP network monitoring, and it's different from round-trip time (RTT) in that it only measures one path from source to destination. Calculating end-to-end latency at a high level necessitates knowing the packet length, connection transmission rate (i.e. bandwidth), and propagation delay, which is generally in the range of 2*10^8 m/s to 3*10^8 m/s.

The end to end delay can be calculated using the following formula

$$x = n * (proc + trans + prop) \quad (2)$$

Where,

x - End to End delay time
n - the number of links between routers
proc - the average processing delay incurred by a router
trans - the average transmission delay
prop - the average propagation delay

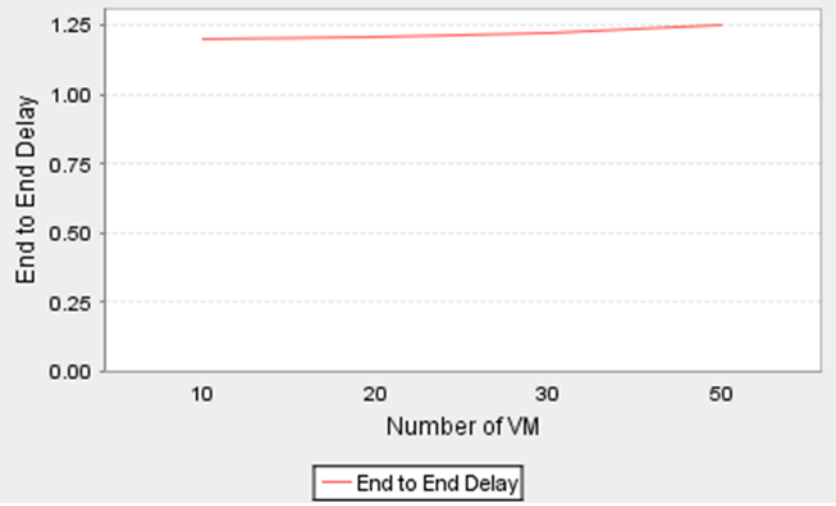

Figure 7: Overall end to end delay of Proposed system

The above-mentioned graph clearly explains the end-to-end delay of the proposed system. The end-to-end delay of the proposed system attains the maximum value of 1.25 and the number of virtual machines is taken from 1 to 50. The above graph depicts that as the end-to-end delay increases the number of VM also increases. The end-to-end delay of the proposed system is calculated by using Escalate Lane Technique in which the network path is maximized.

4.2.3 Execution Time:

The execution time is defined as the product of the number of instructions to the average time per instruction and execution time is calculated using the formula,

$$Execution\ time = Number\ of\ instructions \times average\ time\ per\ instructions \quad (3)$$

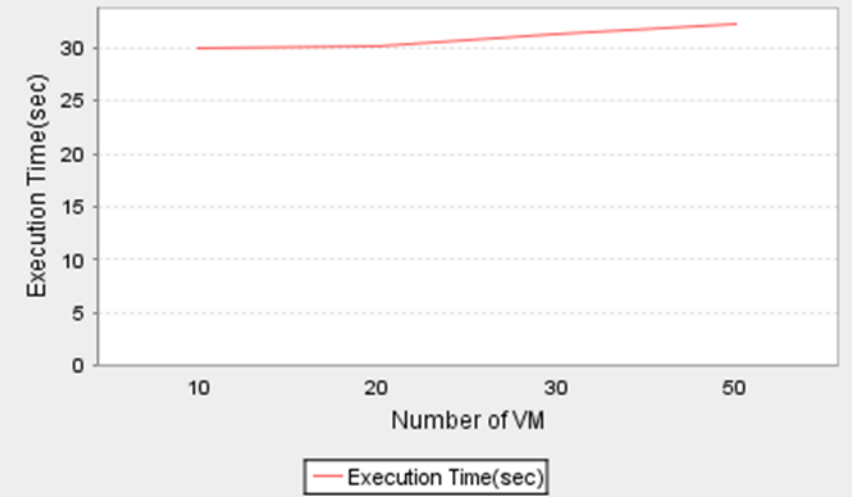

Figure 8: Overall execution time of the proposed system

The above-mentioned graph clearly explains the execution time of the proposed system. The execution time of the proposed system attains the maximum value of 34 seconds and the number of virtual machines is taken from 1 to 50. The above graph depicts that as the execution time increases the number of VM also increases. The execution time of the proposed system is calculated by using Eternal Alacrity in which the availability of virtual machines is increased.

4.2.4 Packet Loss:

When one or more packets of data traveling over a computer network fail to reach their destination, packet loss occurs. Packet loss is caused by either data transmission failures, which are common in wireless networks, or network congestion. Packet loss is expressed as a percentage of packets lost compared to the total number of packets delivered.

$$Packet\ Loss = \frac{Number\ of\ packets\ not\ received}{Total\ number\ of\ packets\ sent} \quad (4)$$

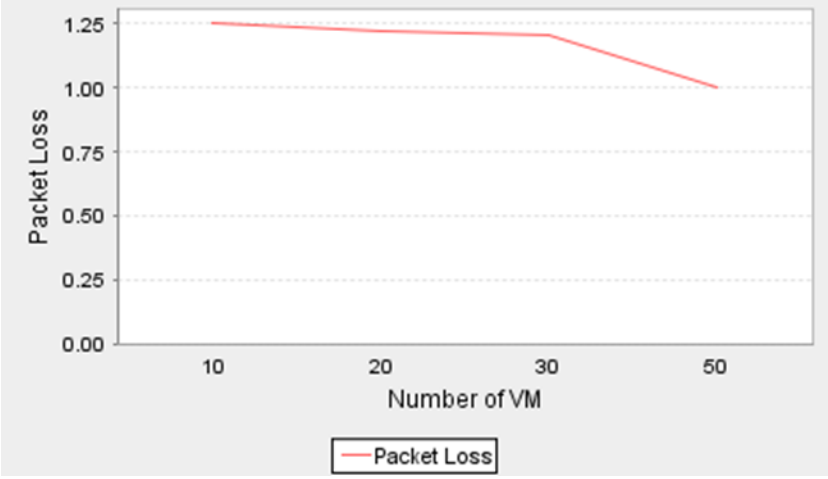

Figure 9: Overall packet loss of the proposed system

The above-mentioned graph clearly explains the packet loss of the proposed system. The packet loss of the proposed system attains the minimum value of 1 and the number of virtual machines is taken from 1 to 50. The above graph depicts that as the number of virtual machines increases the packet loss decreases. The packet loss of the proposed system is calculated by using Eternal Alacrity which stores the information of the process in reserved memory.

4.3 Comparison results of the proposed method:

This section describes the various performance of the proposed method comparing with the results of previous methodologies and depicting their results based on various metrics.

Table 1: Table with execution time comparison

Techniques	Execution time(s)
LXC	80
Docker	85
Docker Machine	90
Proposed	30

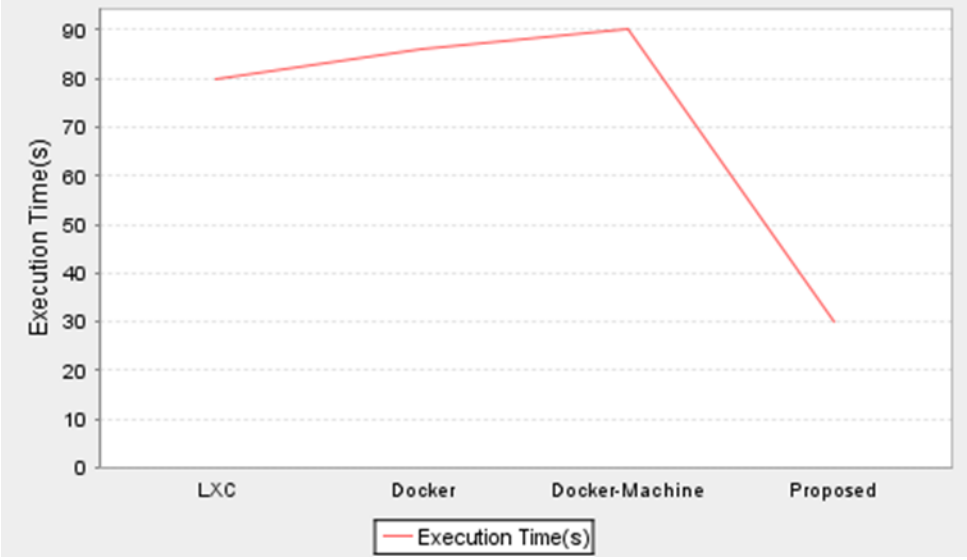

Figure 10: Execution time Comparison

The execution time of the proposed system is compared with the execution time of the various previously proposed techniques. From the graph, it is clear that the execution time of the proposed system is very low that is only 30 seconds than the existing output when compared with LXC [21,22,23], Docker [24,25], and Docker-Machine [26,27], and from the conclusion, it is noted that Docker-Machine has the highest execution time whereas our proposed system has the lowest execution time.

Table 2: Table with IOPS comparison

Techniques	IOPS
LXC	820
Docker	780
Docker Machine	500
Proposed	420

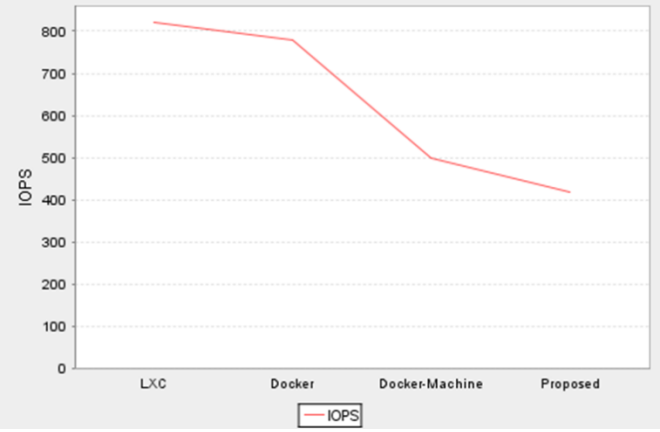

Figure 11: IOPS Comparison

The Input / Output Operations per Second (IOPS) of the proposed system is compared with the Input/output operation per second (IOPS) of the various previously proposed techniques. From the graph, it is clear that the IOPS of the proposed system is very low than the existing output when compared with LXC [21,22,23], Docker [24,25], and Docker-Machine [26,27], and from the comparison, it is noted that LXC has the highest Input/ output operation per second (IOPS) whereas our proposed system has the lowest Input/ output operation per second (IOPS).

Table 3: Table with Round trip time comparison

Techniques	Round trip time (us)
LXC	30
Docker	45
Docker Machine	120
Proposed	140

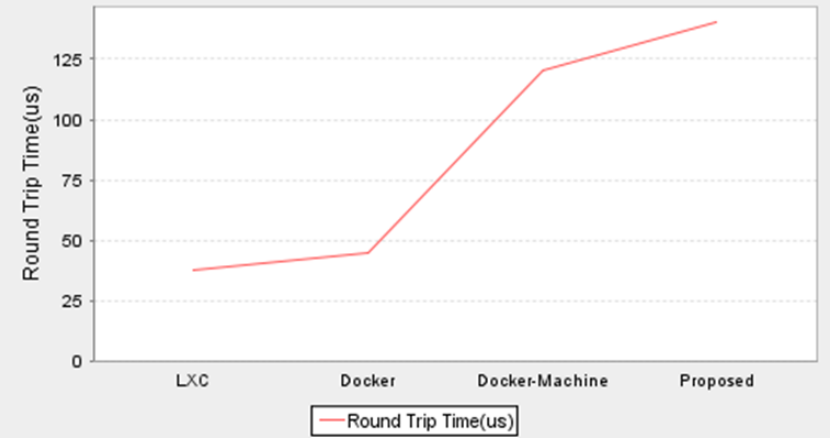

Figure 12: Round Trip Time Comparison

The Round Trip Time of the proposed system is compared with the Round Trip Time of the various previously proposed techniques. From the graph, it is clear that the Round Trip Time of the proposed system is very high that is 140μs than the existing output when compared with LXC [21,22,23], Docker [24,25] and Docker-Machine [26,27] and from the comparison it is noted that LXC has the lowest Round Trip Time whereas our proposed system has the highest Round Trip Time.

Table 4: Table with throughput comparison

Data size (byte)	Throughput (MB/ sec)			
	MAX	IDEAL	ROSI	Proposed
250	3	3	3	13
500	6	6.05	6.05	14
750	7	7.1	7.2	14
1000	7.7	7.8	8	14
1250	8.2	8.4	8.6	14
1500	9.8	10	10.2	14
1750	10.8	10.9	11.2	14
2000	12	12.2	12.4	14

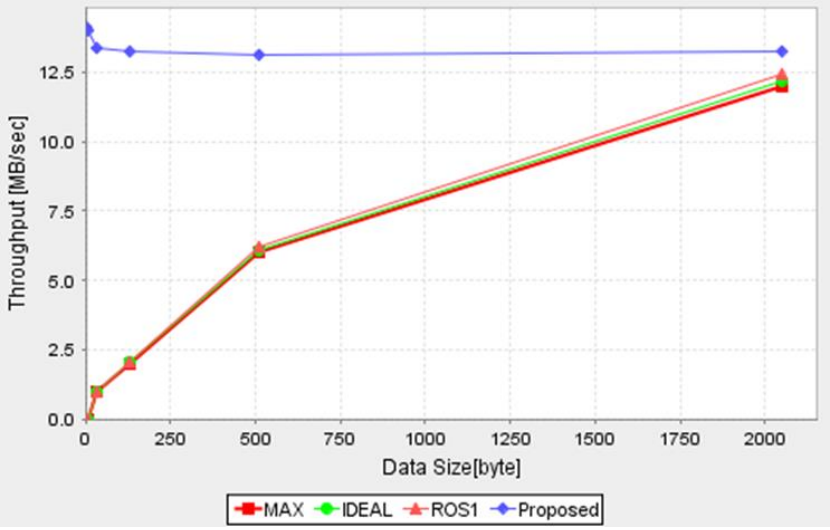

Figure 13: Throughput Comparison

The Throughput of the proposed system is compared with the throughput of the various previously proposed techniques. From the graph, it is clear that the throughput of the proposed system is very high than the existing output when compared with ROS1 [28,29,30,31], and from the comparison, it is noted that ROS1 [28,29,30,31] has the lowest throughput whereas our proposed system has the highest throughput.

5. CONCLUSION:

The technical complexities in achieving fault tolerance in the cloud are removed by using a Ternate Fault Tolerance in Cloud Environment. Compeer Wield, Escalate Lane and Eternal Alacrity provides the solution for major issues like low throughput, loss of virtual machine interaction, minimized network path, and low availability of cloud virtual machines by using photostat to preserve the baseline of every virtual machine to provide high throughput, Ruler node is created to maximize network path and the usage of reserved memory is to provide better availability of the virtual machine. The results of the proposed method are compared with other existing techniques and the proposed Ternate Fault Tolerance in cloud Environment outperforms all the other existing techniques and the proposed system has the highest throughput of 14, high round trip time of 140 μs, and execution time is reduced to 30 seconds and the packet loss is also reduced to 1.

REFERENCE

[1] Sunyaev, A. (2020). Cloud computing. In *Internet computing* (pp. 195-236). Springer, Cham.
[2] Namasudra, Suyel. "Data access control in the cloud computing environment for bioinformatics." *International Journal of Applied Research in Bioinformatics (IJARB)* 11, no. 1 (2021): 40-50.
[3] S. Mustafa, B. Nazir, A. Hayat, A.R. Khan and S.A. Madani, Resource management in cloud computing: Taxonomy,prospects, and challenges, Computers & Electrical Engineering 47 (2015), 186–203lit survey

[4] Irgashevich, Dadamuxamedov Alimjon. "METHODS OF USING CLOUD TECHNOLOGIES IN ISLAMIC EDUCATION INSTITUTIONS." *METHODS* 7, no. 5 (2020).
[5] Pinheiro Junior, Luiz, Maria Alexandra Cunha, Marijn Janssen, and
[6] Ricardo Matheus. "Towards a Framework for Cloud Computing use by Governments: Leaders, Followers and Laggers." In *The 21st Annual International Conference on Digital Government Research*, pp. 155-163. 2020.
[7] Hughes, Kyle, Peter di Pasquale, Alessandra Babuscia, and Lorraine Fesq. "On-demand Command and Control of ASTERIA with Cloud-based Ground Station Services." In *2021 IEEE Aerospace Conference (50100)*, pp. 1-15. IEEE, 2021.
[8] Poulová, Petra, Blanka Klímová, and Martin Švarc. "Cloud Computing in the World and Czech Republic—A Comparative Study." In *Advances in Computer, Communication and Computational Sciences*, pp. 771-778. Springer, Singapore, 2021.
[9] Karthikeyan, L., C. Vijayakumaran, S. Chitra, and Samydurai Arumugam. "Saldeft: Self-adaptive learning differential evolution based optimal physical machine selection for fault tolerance problem in cloud." *Wireless Personal Communications* 118, no. 2 (2021): 1453-1480.
[10] Mohammed, Bashir, Mariam Kiran, Kabiru M. Maiyama, Mumtaz M. Kamala, and Irfan-Ullah Awan. "Failover strategy for fault tolerance in cloud computing environment." *Software: Practice and Experience* 47, no. 9 (2017): 1243-1274.
[11] Sun, PanJun. "Security and privacy protection in cloud computing: Discussions and challenges." *Journal of Network and Computer Applications* 160 (2020): 102642.
[12] Zhang, Peiyun, MengChu Zhou, and Xuelei Wang. "An intelligent optimization method for optimal virtual machine allocation in cloud data centers." *IEEE Transactions on Automation Science and Engineering* 17, no. 4 (2020): 1725-1735.
[13] Kumari, Priti, and Parmeet Kaur. "Topology-aware virtual machine replication for fault tolerance in cloud computing systems." *Multiagent and Grid Systems* 16, no. 2 (2020): 193-206.
[14] Jahanpour, H., H. Barati, and A. Mehranzadeh. "An Energy Efficient Fault Tolerance Technique Based on Load Balancing Algorithm for High-Performance Computing in Cloud Computing." *Journal of Electrical and Computer Engineering Innovations (JECEI)* 8, no. 2 (2020): 169-182.
[15] Attallah, Salma MA, Magda B. Fayek, Salwa M. Nassar, and Elsayed E. Hemayed. "Proactive load balancing fault tolerance algorithm in cloud computing." *Concurrency and Computation: Practice and Experience* 33, no. 10 (2021): e6172.
[16] Gonzalez, Christopher, and Bin Tang. "FT-VMP: Fault-Tolerant Virtual Machine Placement in Cloud Data Centers." In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1-9. IEEE, 2020.
[17] Zhou, Ao, Qibo Sun, and Jinglin Li. "Enhancing reliability via checkpointing in cloud computing systems." *China Communications* 14, no. 7 (2017): 1-10.
[18] Chen, Rong, and Haibo Chen. "Asymmetric virtual machine replication for low latency and high available service." *Science China Information Sciences* 61, no. 9 (2018): 1-15.
[19] Rezaeipanah, Amin, Musa Mojarad, and Ahad Fakhari. "Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic." *International Journal of Computers and Applications* (2020): 1-9.

[20] Khaldi, Miloud, Mohammed Rebbah, Boudjelal Meftah, and Omar Smail. "Fault tolerance for a scientific workflow system in a cloud computing environment." *International Journal of Computers and Applications* 42, no. 7 (2020): 705-714.
[21] Ray, Benay, Avirup Saha, Sunirmal Khatua, and Sarbani Roy. "Proactive fault-tolerance technique to enhance reliability of cloud service in cloud federation environment." *IEEE Transactions on Cloud Computing* (2020).
[22] Ruan, B., Huang, H., Wu, S., & Jin, H. (2016, November). A performance study of containers in cloud environment. In *Asia-Pacific Services Computing Conference* (pp. 343-356). Springer, Cham.
[23] Hsieh, Madison, and Kevin Fox. *Product Consistency Test Results for the LXC-Series Glasses*. No. SRNL-STI-2020-00215. Savannah River Site (SRS), Aiken, SC (United States); SRNL, 2020.
[24] Mardan, Asraa Abdulrazak Ali, and Kenji Kono. "When the virtual machine wins over the container: Dbms performance and isolation in virtualized environments." *Journal of Information Processing* 28 (2020): 369-377.
[25] Moravcik, Marek, et al. "Comparison of LXC and Docker Technologies." *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. IEEE, 2020.
[26] Putri, Adinda Riztia, Rendy Munadi, and Ridha Muldina Negara. "Performance analysis of multi services on container Docker, LXC, and LXD." *Bulletin of Electrical Engineering and Informatics* 9.5 (2020): 2008-2011.
[27] Haque, Mubin Ul, Leonardo Horn Iwaya, and M. Ali Babar. "Challenges in docker development: A large-scale study using stack overflow." *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2020.
[28] Chiang, Ron C. "Contention-aware container placement strategy for docker swarm with machine learning based clustering algorithms." *Cluster Computing* (2020): 1-11.
[29] Drilon, Alexander, et al. "ROS1-dependent cancers—biology, diagnostics and therapeutics." *Nature reviews Clinical oncology* 18.1 (2021): 35-55.
[30] Yun, M.R., Kim, D.H., Kim, S.Y., Joo, H.S., Lee, Y.W., Choi, H.M., Park, C.W., Heo, S.G., Kang, H.N., Lee, S.S. and Schoenfeld, A.J., 2020. Repotrectinib Exhibits Potent Antitumor Activity in Treatment-Naïve and Solvent-Front–Mutant ROS1-Rearranged Non–Small Cell Lung Cancer. *Clinical Cancer Research*, *26*(13), pp.3287-3295.
[31] Lin, Jessica J., Noura J. Choudhury, Satoshi Yoda, Viola W. Zhu, Ted W. Johnson, Ramin Sakhtemani, Ibiayi Dagogo-Jack et al. "Spectrum of Mechanisms of Resistance to Crizotinib and Lorlatinib in ROS1 Fusion–Positive Lung Cancer." *Clinical Cancer Research* 27, no. 10 (2021): 2899-2909.
[32] Sato, Hiroki, Adam J. Schoenfeld, Evan Siau, Yue Christine Lu, Huichun Tai, Ken Suzawa, Daisuke Kubota et al. "MAPK pathway alterations correlate with poor survival and drive resistance to therapy in patients with lung cancers driven by ROS1 fusions." *Clinical Cancer Research* 26, no. 12 (2020): 2932-2945.
[33] Maruyama, Yuya, Shinpei Kato, and Takuya Azumi. "Exploring the performance of ROS2." *Proceedings of the 13th International Conference on Embedded Software*. 2016.