Proactive Load Balance Fault Tolerance in Real Cloud Computing (PLBFT)

Presented By

Salma Mohamed Ali Attallah

Supervised By

Prof. Salwa Nassar **Prof. Elsayed Hemayed** **Prof. Magda Fayek**

Outlines

- Introduction
- Background
- Literature Review
- Proposed Model
- Experimental Results
- Conclusion & Future Work

Introduction

- **Motivation:**
  - Cloud computing is one of the most scientific revolutions that provide internet service on demand.
  - Cloud computing systems may have a variety of faults, such as hardware failure, software failure, abuse of customers, etc.
  - Fault tolerance design is one of the most critical model for cloud systems.
  - Fault tolerance is responsible for saving the system from errors to achieve high cloud service efficiency and reliability.
  - Some techniques of fault tolerance are **reactive techniques** that attempt to recover the system after failures, while others are **proactive techniques** that predict failure before it occurs to save as much as possible of the available system.

Introduction – Cont.

- **Contribution:**
  - The main objective of this thesis is to propose a new model of proactive fault tolerance.
  - We implemented six different fault tolerance policies separately in addition to the technique of load balancing and are applied to a private cloud computing system.
  - The proposed model achieved better results when compared to two proactive fault tolerance models **Adaptive Fault Tolerance in Real-Time Cloud (AFTRC)** [1] and **FT-Cloud** [2].

Outlines

- Introduction
- Background
- Literature Review
- Proposed Model
- Experimental Results
- Conclusion & Future Work

Fault Tolerance

- Fault tolerance technique is a technique of failover which allows the device to continue its operation even if a failure has occurred.
- The key function of the fault tolerance technique is to detect any fault occurring in components of each device, such as memory, disk storage, network, or computer processing unit (CPU).
- The value of the technique of fault tolerance is to keep the system up and running which achieves system high availability and reliability.
- Cloud computing providers aim to develop their system by implementing fault tolerance techniques to achieve high availability systems.

Fault Tolerance

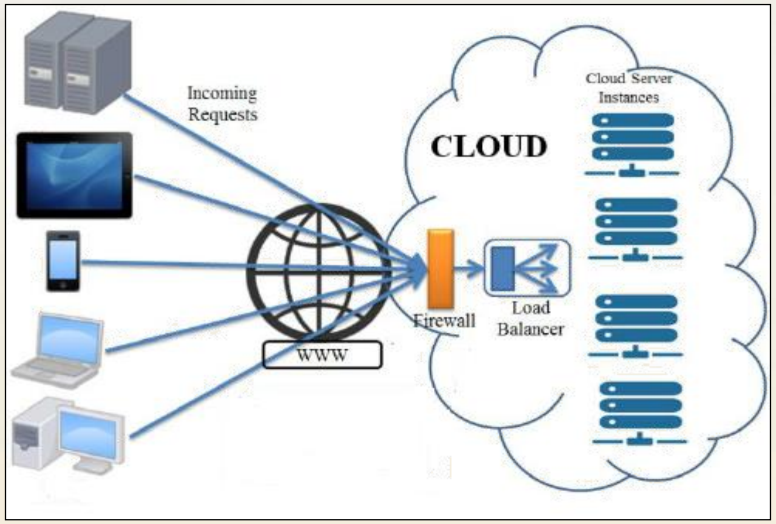
- The failures in cloud computing can be divided into two classes:
  - Data failures (e.g., data theft, missing source data).
  - Failure of hardware such as defective or slow VMs, and exception of storage access

- There are primarily two common fault tolerant policies
  - Proactive Fault Tolerance Policy
  - Reactive Fault Tolerant Policy

Load Balancing

- Load balancing in cloud computing distributes workload across various computing resources, such as migrating virtual machines.

- In general, load balancing is aimed at:
  - Maximizing the use of resources
  - Decreasing reaction time
  - Preventing any single resource overload.



Outlines

Literature Review - Fault Tolerance Techniques

- **Most common popular proactive FT techniques are:**

I. Adaptive Fault Tolerance in Real Time Cloud (AFTRC) [1]:

- It relies on running same application that is implemented with different algorithms on different virtual machines.
- It checks the time for each run and measured all virtual machine reliability.

II. FT-Cloud [2]:

- In FT-Cloud, three well-known fault tolerance strategies are introduced:
  - **Recovery Block (RB)**
  - **N-Version Programming (NVP)**
  - **Parallel**

[illegible]

Literature Review - Load Balancing Techniques

- **Load Balancing with Fault Tolerance [3]**:
  - It is a new modified Round Robin algorithm that is evaluated on a cloud simulator
  - The modified Round Robin algorithm allocates the VM to the host in a cyclical fashion
  - At the initial allocation stage, the Modified Round Robin Algorithm retains the fault tolerance level of the services themselves.

- **Preference Based Fault Management (PBFM) [4]:**
  - Cloud federation allows one cloud service provider (CSP) to offer their unutilized computing resource (VMs) to other CSP when their demand is low or enables them to outsource the resources from different CSP whenever there is high demand.

Outlines

- Introduction
- Background
- Literature Review
- Proposed Model
- Experimental Results
- Conclusion & Future Work

Proposed Model

- We are introducing a new proactive load balancing model for fault tolerance within a real cloud system.

- This model aims to increase the efficiency of cloud services by minimizing as much time as possible for cloud breakdowns.

- The virtual machine CPU has been reported as the most common cause of a significant failure.

Proposed Model - Alarms

- OpenStack is an open source cloud computing infrastructure software.
- From the survey, we found openstack has a telemetry service that called “**Ceilometer**”, it used for billing, resource monitoring, and alarming capabilities.
- Two alarm are initiated as follows:
  - **Warning Alarm**, initiated when CPU utilization equal 70%.
  - **Error Alarm**, initiated when CPU utilization equal 80% & 90%.

* (A CPU bottleneck appears if its utilization goes beyond 80% for a sustained period of time [E. Ciliendo and T. Kunimasa, Linux Performance and Tuning Guidelines, 1st ed. ibm.com/redbooks, Jul. 2007])

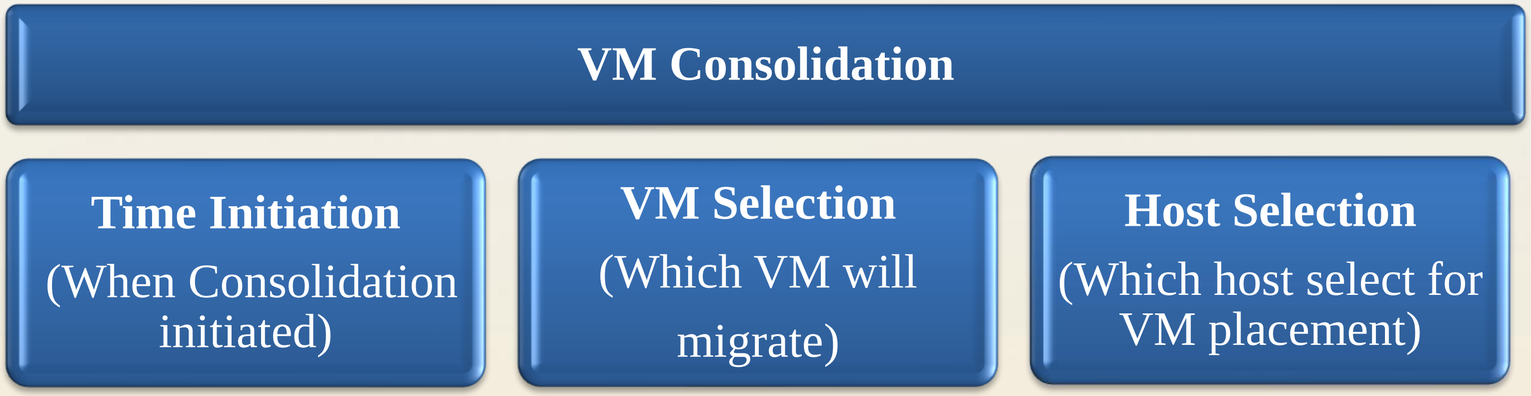
Proposed Model - Archiving Database

- We also developed a MySQL database that is designed to save alarm specifications and record all device faults.

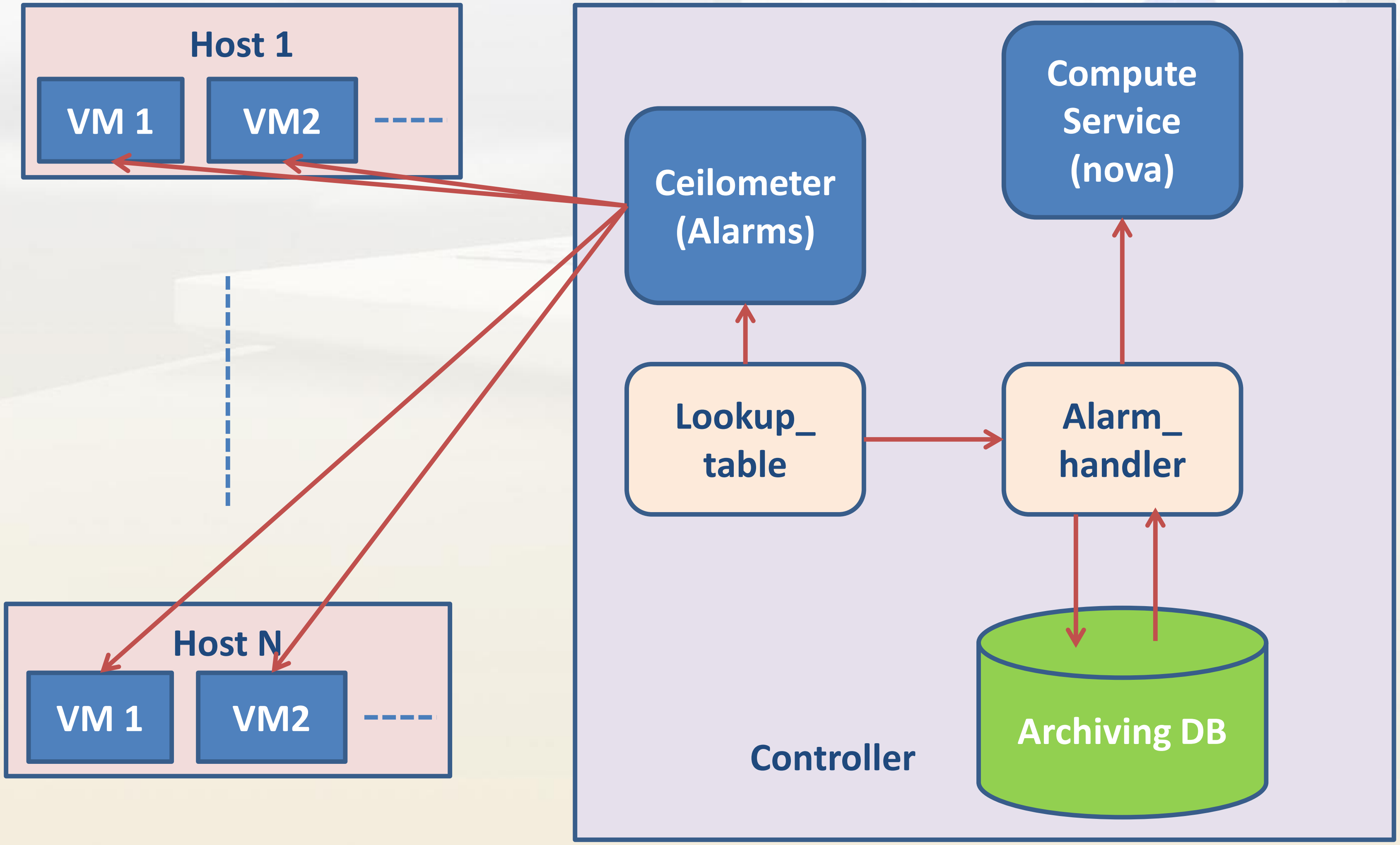
- It primarily includes two "**vm_alarms**" and "**vm_faults**" tables:
  - **vm_alarms:**
    - It has all the alarm data, such as the name of the virtual machine, the alarm threshold, etc.
  - **vm_faults:**
    - This covers all virtual machine bugs that have existed in the system previously.
    - It can be called proposed model algorithm logs.

Proposed Model – VM Consolidation

- Consolidation and migration of Virtual Machines increase system utilization, availability, fault tolerance, and energy efficiency.
- The main objective of VM consolidation is to connect some virtual cloud machines to a specific host through live migration of virtual machines.



Proposed Model - Model Architecture

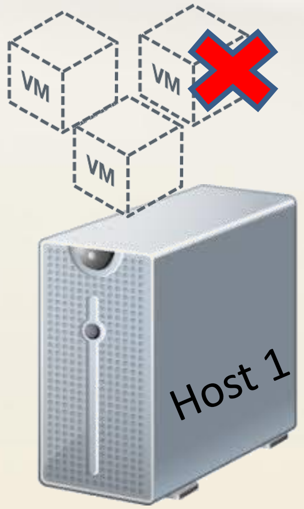


Proposed Model - Applied Policies

1) Rebooting the defective virtual machine policy:

– One of the most commonly used fault tolerance policies is to reboot (restart) a defective virtual machine.

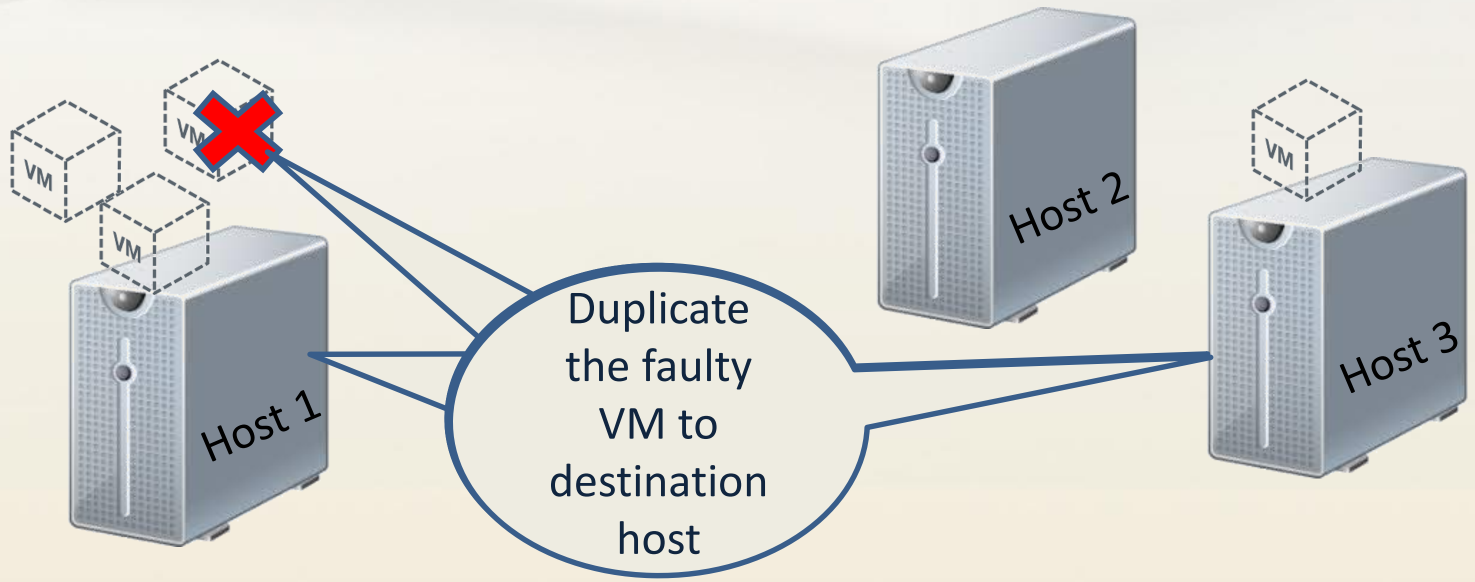
– To ensure the status of a virtual machine is reset to its initial state.



Proposed Model - Applied Policies

2) Duplicating the defective VM to another host policy:

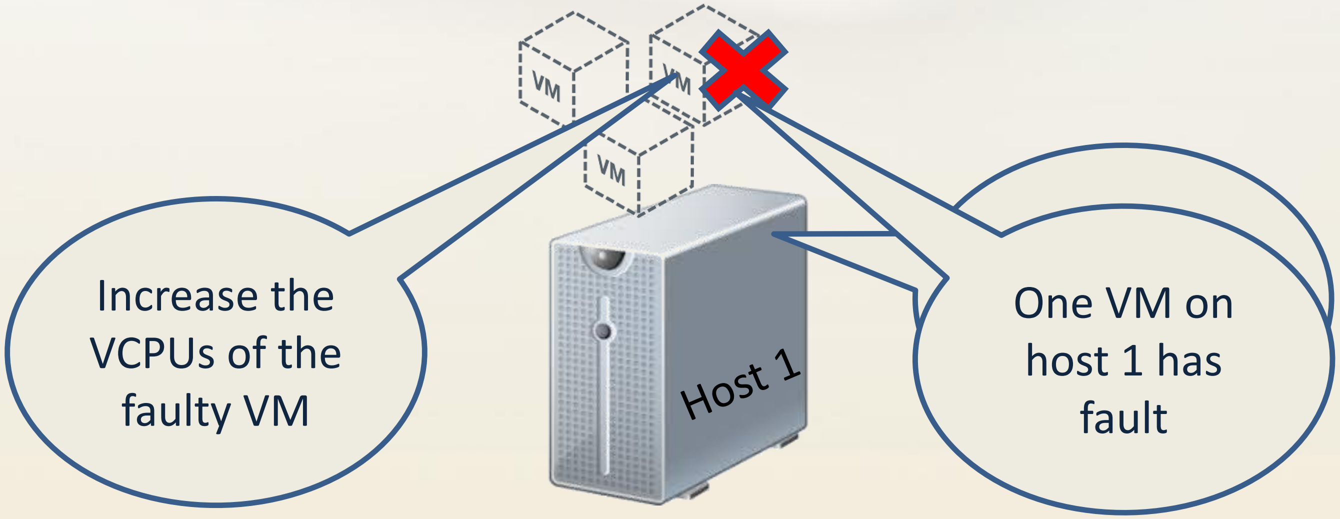
- The faulty VM is duplicated (copied) to another host.
- The destination host is selected randomly.
- The purpose of this policy is to keep two VMs running on a cloud system that acts as a VM backup



Proposed Model - Applied Policies

3) Scaling up the number of virtual CPUs allocated to a defective VM policy:

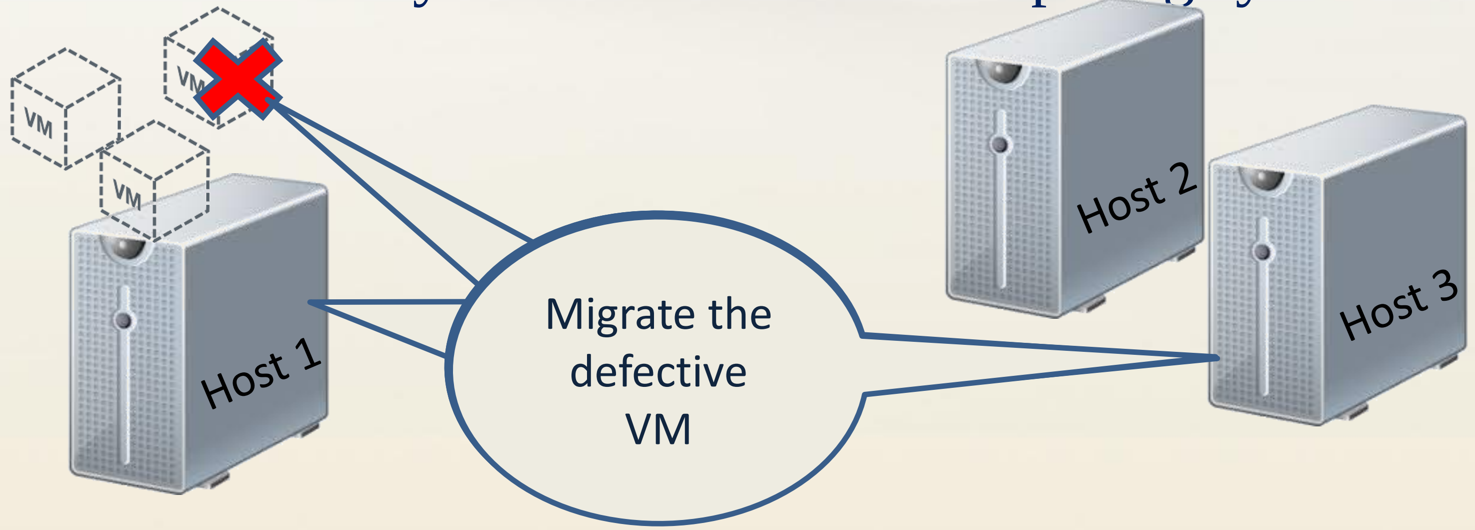
– One of the constructive fault tolerance policies is to scale up (increase) the allocated virtual CPUs to the faulty virtual machine.


VM
VM
VM
Host 1
Increase the VCPUs of the faulty VM
One VM on host 1 has fault

Proposed Model - Applied Policies

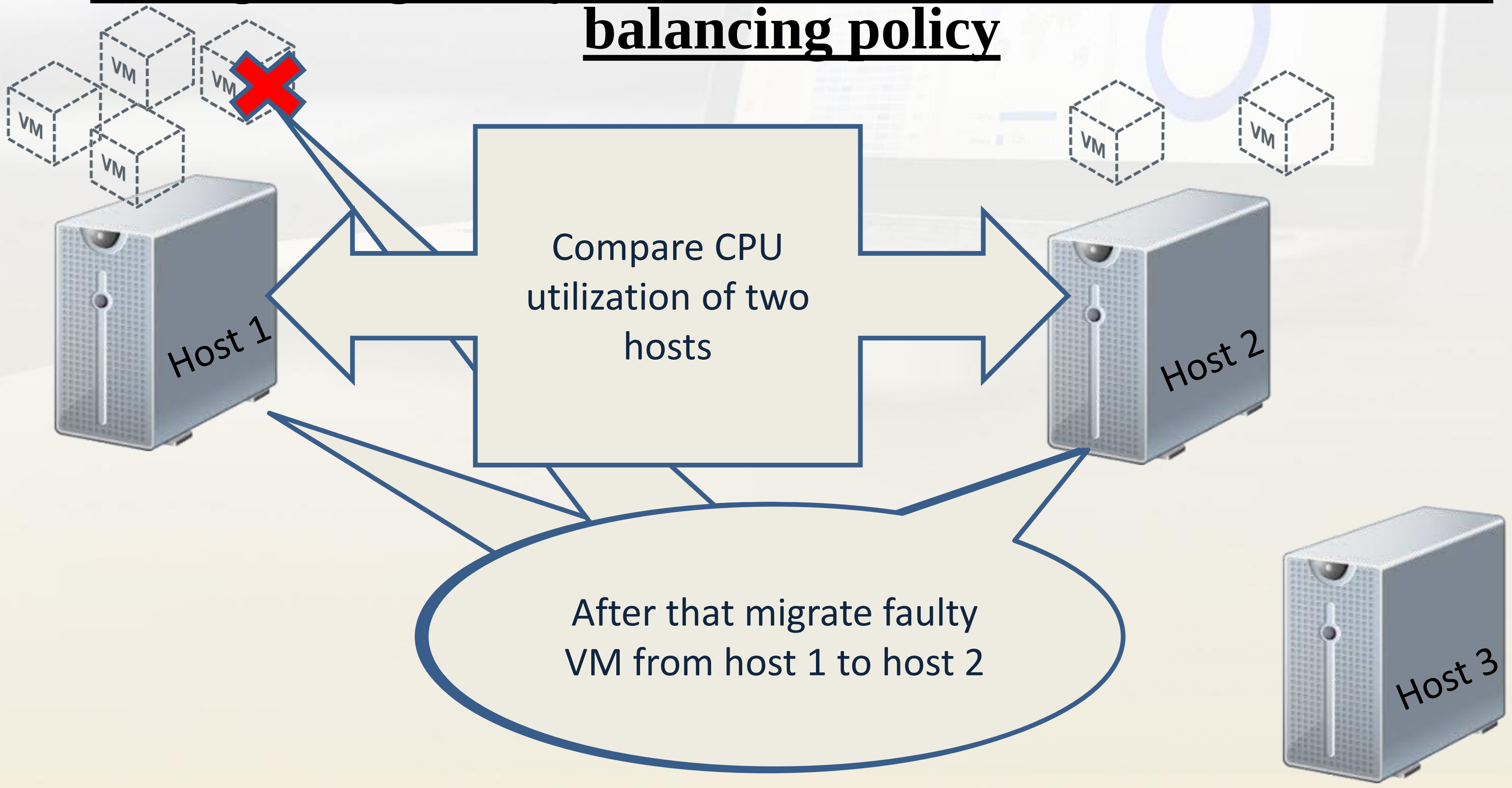
4) Migrating the defective VMs to a random host policy:

- Virtual machine migration refers to transferring the faulty virtual machine to new host.
- We implemented our model with a live migration mechanism because of the increasing system availability and reliability factors of a cloud computing system.



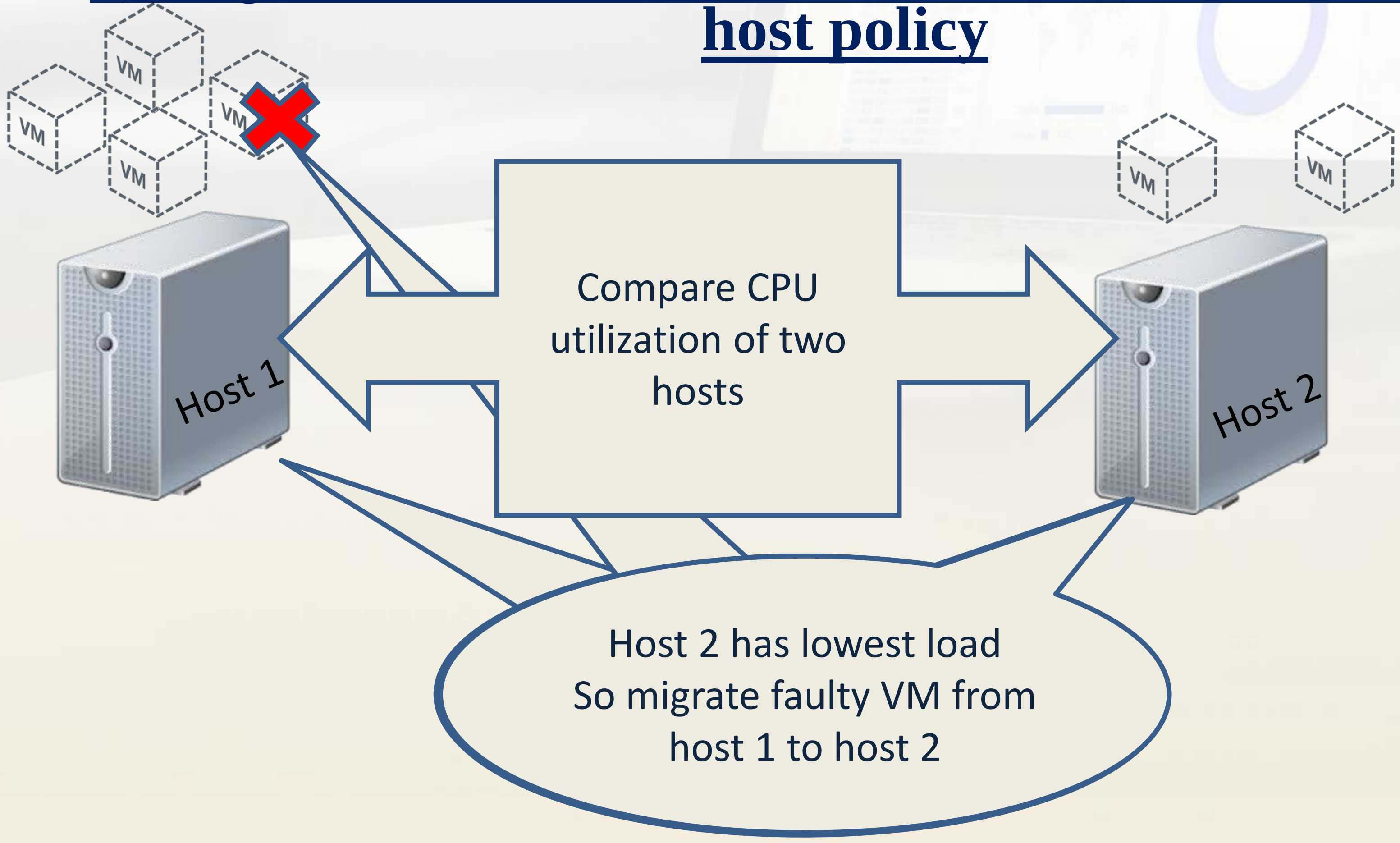
Proposed Model - Applied Policies

5) Migrating faulty VMs to random host with host load balancing policy


VM
VM
VM
VM
VM
Host 1
Compare CPU utilization of two hosts
VM
VM
Host 2
After that migrate faulty VM from host 1 to host 2
Host 3

Proposed Model - Applied Policies

6) Migration of the defective VM to the minimum loaded host policy



VM
VM
VM
VM
VM
VM
Host 1
Host 2
Compare CPU utilization of two hosts
Host 2 has lowest load So migrate faulty VM from host 1 to host 2

Outlines

- Introduction
- Background
- Literature Review
- Proposed Model
- Experimental Results
- Conclusion & Future Work

Experimental Results

- We have implemented the proposed model by using a real cloud computing infrastructure.
- The preliminary experiment of the proposed algorithm indicates a consequent decrease in failure rates.

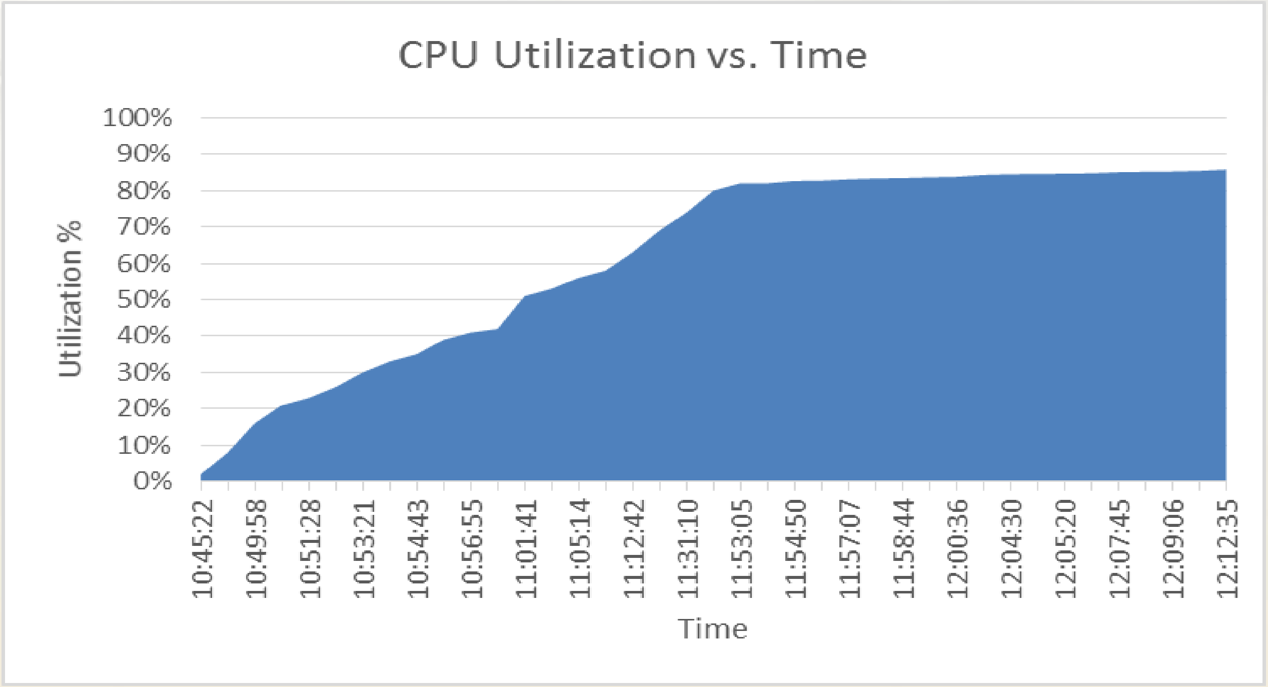
- **Testing Environment:**
  - The proposed model was tested on a real cloud computing infrastructure, with five nodes and three virtual machines launched for testing.
    - **The cloud controller node** is the management node in the cloud.
    - **The network node** is in charge of all networks inside the cloud system.
    - All virtual computers in the cloud computing environment are located in **three computing nodes**.

Experimental Results
CPU Threshold Analysis

- CPU utilization threshold is our factor for generating alarms of the faulty virtual machine.
- Each virtual machine has various loads that differ in applications and users usage.
- So one of our contribution is analyzing thresholds of CPU utilization to get the most better threshold should fault tolerance model run at it.
- During searching for performance tool, we found **Stress** that is a widely used tool in performance tests and it stresses the CPU literally [8].

Experimental Results
CPU Threshold Analysis

- **Stress-ng** is a new generation of stress tool that has been built with new features in mind.
- During the **stress-ng** run, we kept track of the CPU utilization ratio for over an hour and a half. Over time, the CPU utilization increased, as shown in the graph below.



Experimental Results Measurement Metrics

1) MTBF (Mean Time Between Failures) is the arithmetic mean of the times between random failures of a component or system, and formula is: **MTBF = MTTF + MTTR [6,7]**
2) MTTF is the expected time to failure occurrence while the system was being operational up, its formula is:

$$\mathbf{MTTF = T/N}$$

(Where T = total time and N = Number of units under test)

3) MTTR (Mean Time To Repair) is the expected time to repair the failure.
4) The failure rate (λ) is a measurement predictor of the system's availability and is the opposite of the MTBF:

$$\boldsymbol{\lambda = 1/\mathrm{MTBF}} \qquad \textbf{[7]}$$

Experimental Results Measurement Metrics

5) The probability of the system (P(t)) will work for a time (t) without failure is the **reliability** and its formula is:

$$\mathbf{P(t) = e^{-t/MTBF} = e^{-\lambda t}} \qquad [6]$$

6) Failure probability of a component is its failure rate. Once the failure rate is defined, the failure probability FP(t), follows:

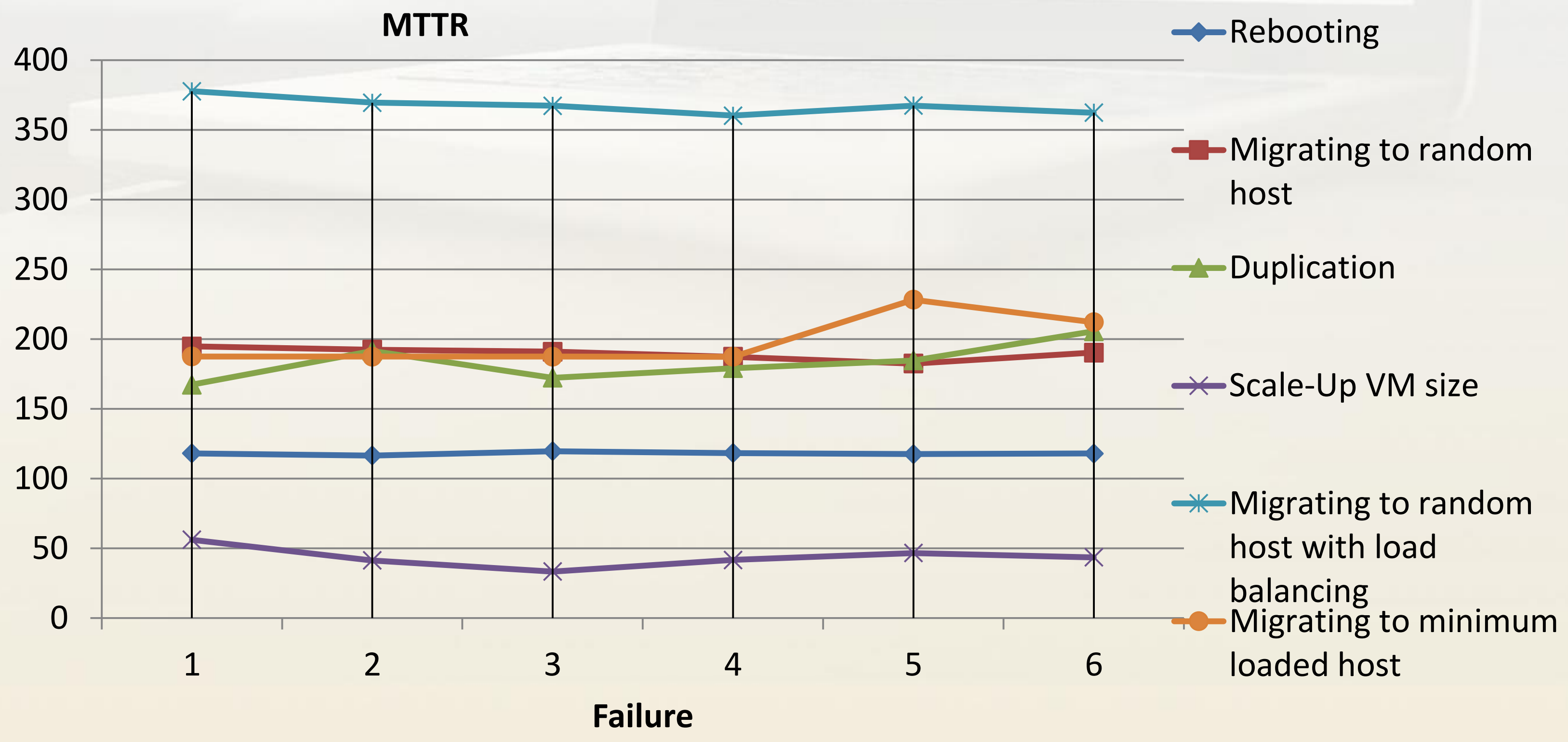
$$\mathbf{FP(t) = 1- e^{-\lambda t}} \qquad [6]$$

7) Availability refers to the percentage of time when the system is operational.

$$\textbf{Availability = MTBF / (MTBF+MTTR)} \ [7]$$

Experimental Results
Performance Measures

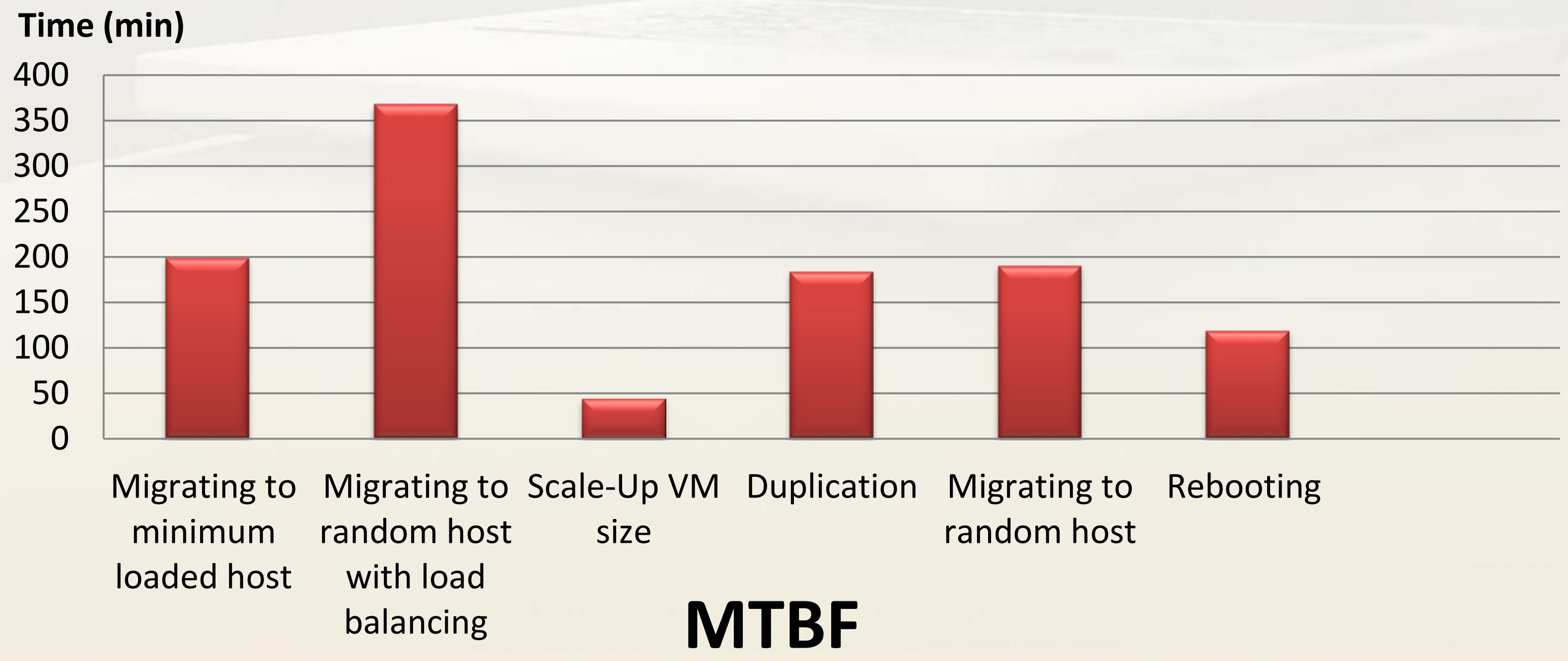
- The following figure depicts the Mean Time To Repair (MTTR) collected from six different policy areas.



Experimental Results Performance Measures

- The outcomes of six related MTBF policies are depicted in the following figure.

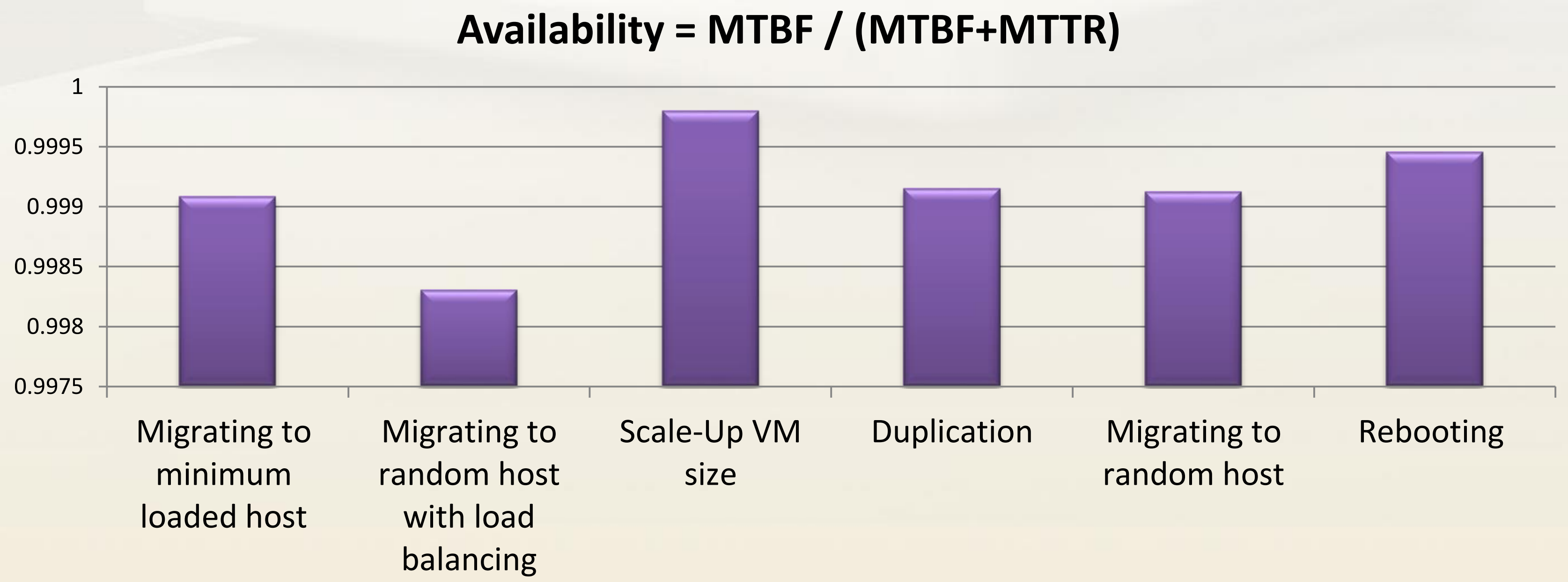
MTBF = MTTF + MTTR

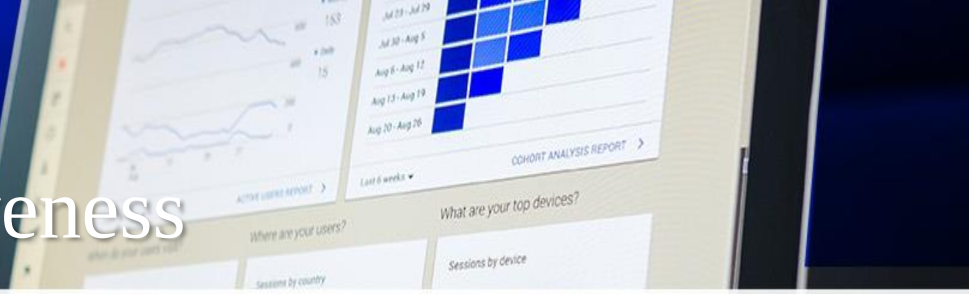


Experimental Results
Load Balancing Effectiveness

- After testing the proposed six policies for five days and tracking them with two virtual machines.

- We measure the availability of the cloud system as shown below





Experimental Results
Load Balancing Effectiveness

- The difference in availability values in six policies is small.
- Downtime is a better way to consider availability [6,7].

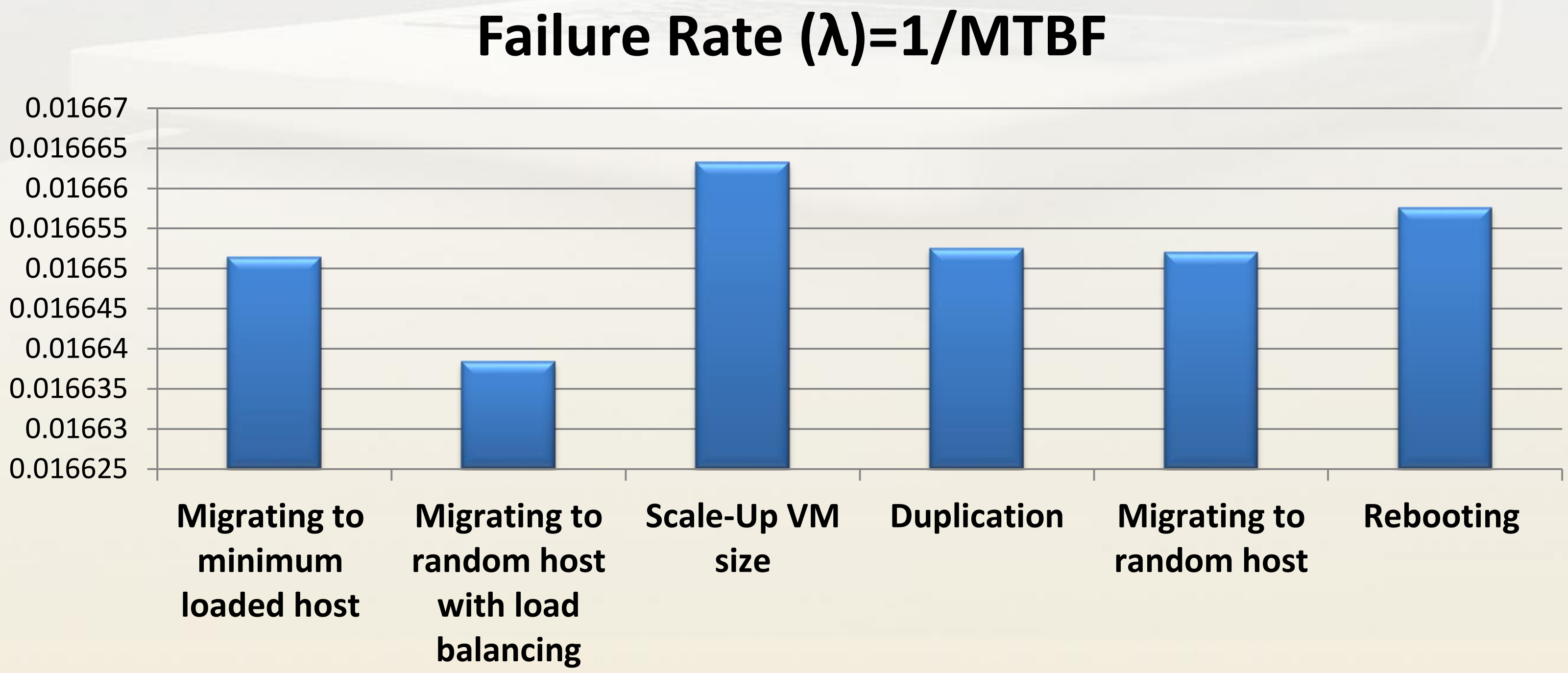
$$\mathbf{Downtime} = \frac{\mathbf{Uptime}}{\mathbf{availability}} - \mathbf{Uptime}$$

- The following table compares the availability of each policy and the associated downtime.

Policy	Availability	Downtime
Rebooting	99.95%	4.38 hours/year
Migrating to a random host	99.91%	7.884 hours/year
Duplication	99.92%	7 hours/year
Scale-Up VM size	99.97%	2.628 hours/year
Migrating to random host with load balancing	99.83%	14.892 hours/year
Migrating to minimum loaded host	99.9%	8.76 hours/year

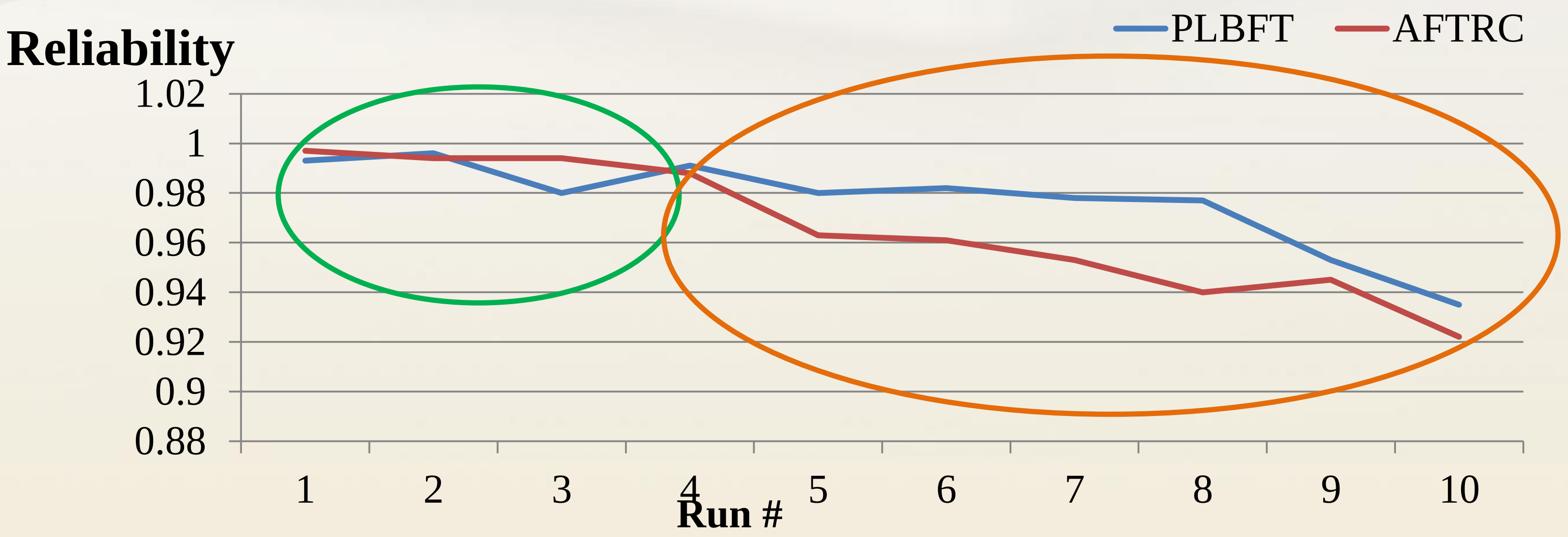
Experimental Results
Load Balancing Effectiveness

- After five days of monitoring the cloud system.
- We measured the failure rate of the proposed six policies.



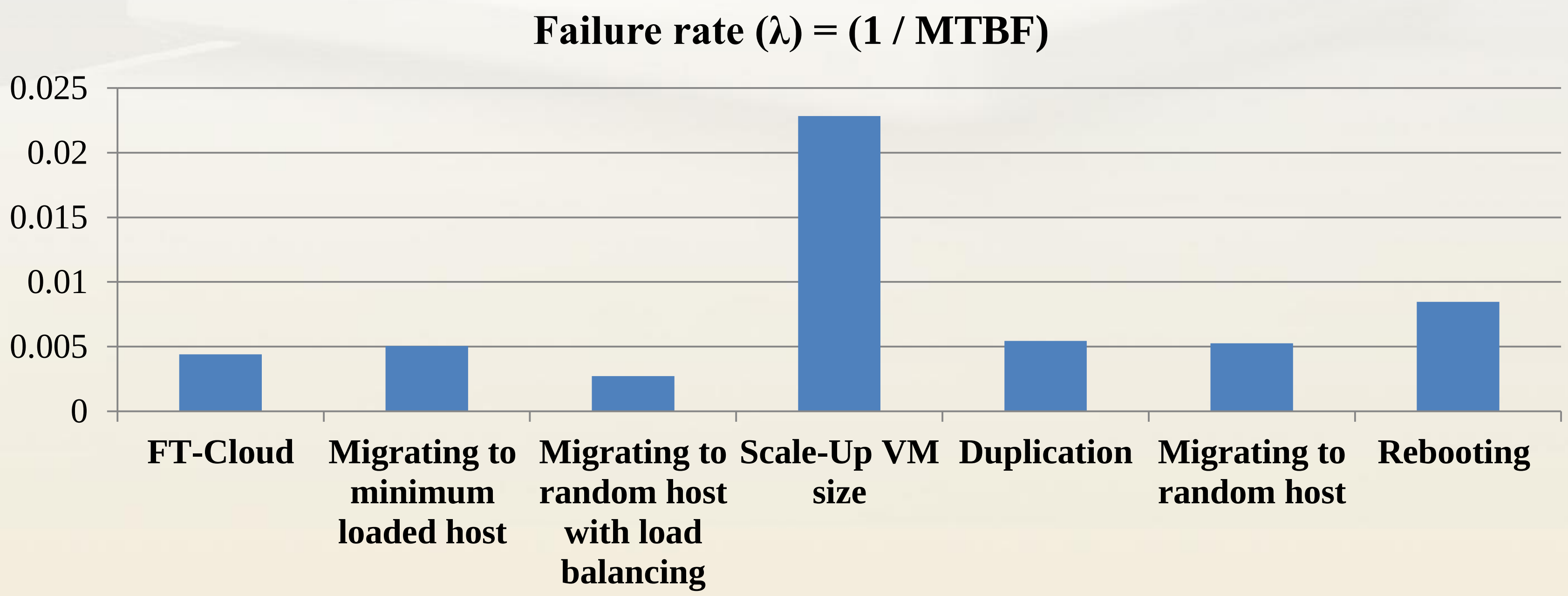
Experimental Results
Proposed Model vs. AFTRC

- The proposed model was evaluated in the same setting as the AFTRC model implemented in [1].
- And we have measured the average reliability of the proposed model. $\mathbf{R(t) = e^{-\lambda t}}$



Experimental Results Proposed Model vs. FT-Cloud

- The defective VM has been replicated to several cloud hosts, and all copies are running at the same time.
- The key drawback of this algorithm is resource waste.
- We calculated the failure rate for the proposed model and FT-Cloud



Experimental Results
Proposed Model vs. FT-Cloud

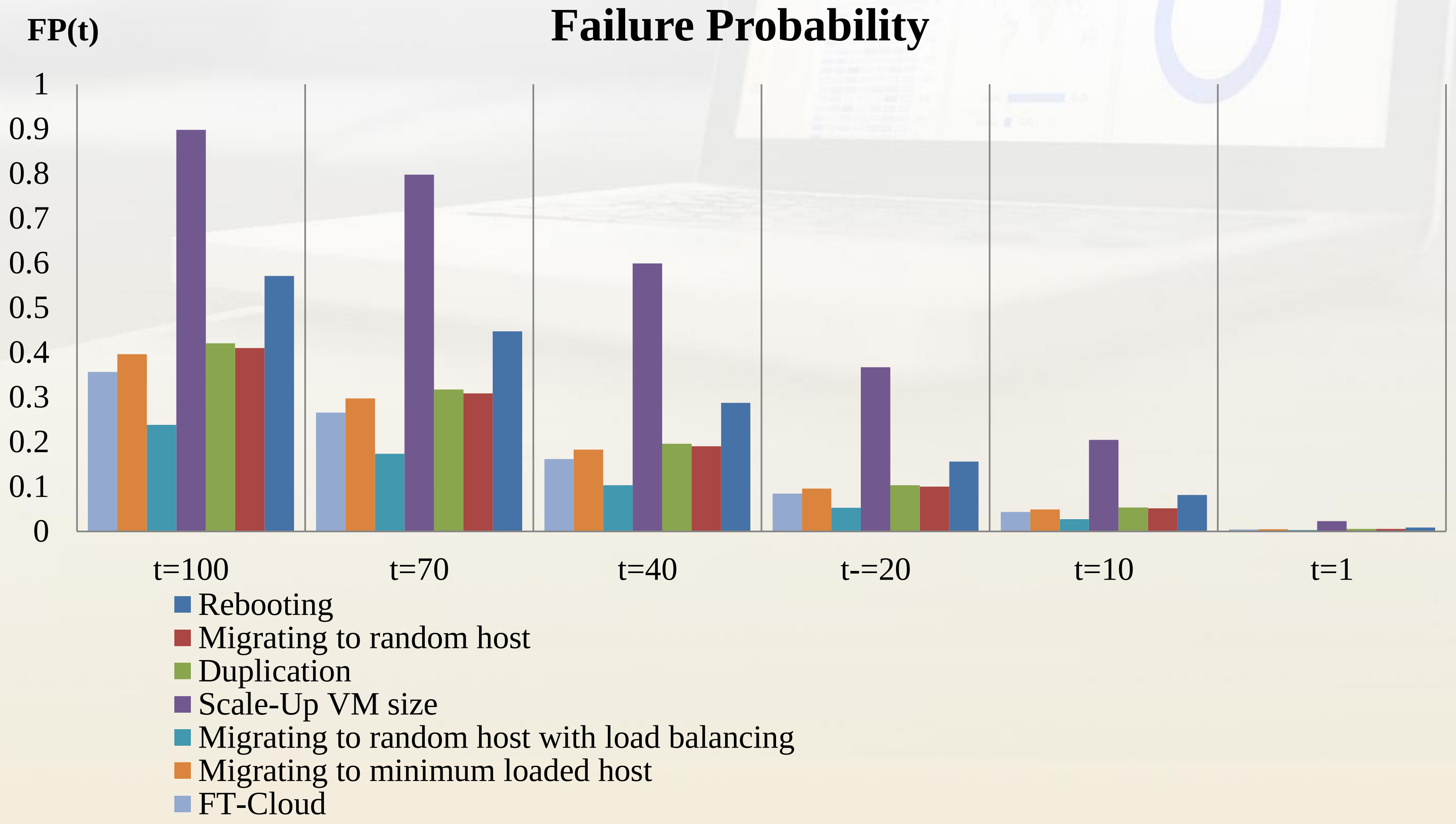
- When we calculated the failure probability for the proposed model and FT-Cloud algorithm.

$$\mathbf{FP(t)= 1-} \, e^{-\lambda t}$$

(Where t is the time by hours, and λ is the failure rate)

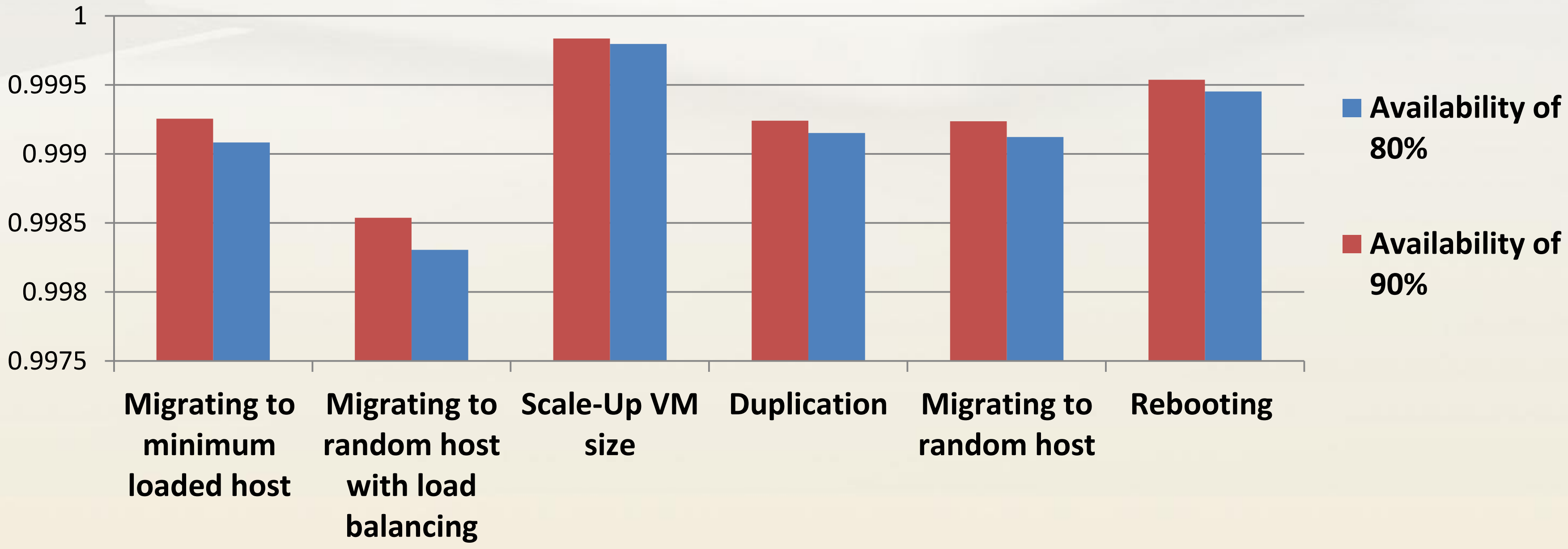
- We discovered that migrating to a random host with a load balancing strategy had the lowest failure probability at different time t, as shown in figure

Experimental Results
Proposed Model vs. FT-Cloud



Experimental Results
Alarm Trigger Variation

- The figure below depicts the availability of the cloud system when the error thresholds are set to 80% and 90%, respectively.



Outlines

- Introduction
- Background
- Literature Review
- Proposed Model
- Experimental Results
- Conclusion & Future Work

Conclusion

- A new proactive fault tolerance model with load balancing for real cloud computing infrastructure is proposed in this study.
- The proposed model handles CPU faults in VMs proactively, using six different strategies to prevent the cloud system from dropping.
- By comparing the proposed model with the AFTRC model, the proposed model achieved average reliability of 0.9765 unless AFTRC achieved 0.9657.
- By implementing the FT-Cloud algorithm and comparing it with the proposed model, the proposed model achieved minimum failure rate and minimum failure probability in load balancing policy.

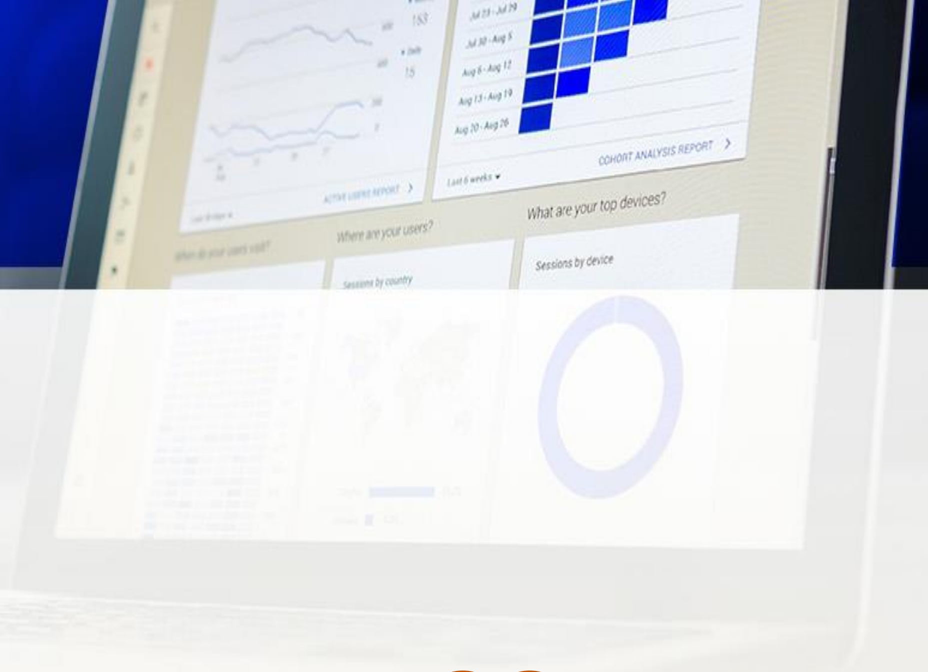
Future Work

- We hope implement new policy that migrate the faulty VM to random host that determined as one of minimum loaded set of hosts.

- We hope to introduce a multiple fault model in the future that deals with various types of faults such as memory faults and storage errors

References

1. Malik, S., and Huet, F., 2011, "Adaptive Fault Tolerance in Real Time Cloud Computing." In Services (SERVICES), 2011 IEEE World Congress on, pp. 280-287
2. Zheng, Z., Zhou, T. C., Lyu, M. R., and King, I., “FTCloud: A Component Ranking Framework for Fault-Tolerant Cloud Applications” 21st International Symposium on Software Reliability Engineering, IEEE, 2010, pp. 398-407.
3. Gayathri, G., Latha, R., 2017, “Implementing a Fault Tolerance Enabled Load Balancing Algorithm in the Cloud Computing Environment”, International Journal of Engineering Development and Research, Volume 5, Issue 1, pp. 249-256.
4. Ray, B., Saha, A., Khatua, S., & Roy, S., 2020, "Proactive Fault-Tolerance Technique to Enhance Reliability of Cloud Service in Cloud Federation Environment". IEEE Transactions on Cloud Computing, Advance online publication. doi:10.1109/TCC.2020.2968522
5. Lagwal, M., Bhardwaj, N., 2017, “Load balancing in Cloud Computing using Genetic Algorithm”, International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 560-565.
6. https://www.fiixsoftware.com/how-do-maintainability-and-reliability-affect-availability/
7. IEEE Recommended Practice for the Use of Probability Methods for Conducting a Reliability Analysis of Industrial and Commercial Power Systems (https://ieeexplore.ieee.org/document/7034995?denied=)
8. H. Cheng, Z. Chen, N. Sun, F. Chen, and M. Wang, “Evaluation framework of virtualization systems for cloud computing,” in 2012 IEEE Asia Pacific Cloud Computing Congress (APCloudCC), 2012



Any Question??

THANKS