Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/108024, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Hybrid Fault Tolerant Scheme to Manage VM Failure in the Cloud

P. Padmakumari*, A. Umamakeswari and M. Akshaya

School of Computing, SASTRA University, Thanjavur - 613401, Tamil Nadu, India;
padmalec.sastra@cse.sastra.edu, aum@cse.sastra.edu, imsunitha@gmail.com

Abstract

Objectives: To managing the Virtual Machine and improve the reliability and availability of the resources using Hybrid fault tolerance method. **Methods/Statistical Analysis**: Basically two main fault tolerant policies namely reactive fault tolerance and proactive fault tolerance. Reactive fault tolerant techniques attempts to reduce the effort required for recovery after a failure, whereas a proactive fault tolerant technique attempts to prevent the occurrence of a fault. The existing method aims to prevent the occurrence of VM failure using a proactive fault tolerant technique which is a static method. But it provides no solution for VM failure, which might occur while the application is still running. The proposed paper ensures reliability of VM by considering both the reactive and proactive techniques in hybrid ways of tolerating the VM failure. In the proposed system, two types of VM failure, insufficient storage and variable capacity change are detected. These failures are tolerated using a self healing mechanism implemented at the host level. **Result:** The proposed scheme is implemented using Cloudsim simulator, in order to highlight the performance of how VM can be managed. **Applications:** Virtual Machine failure can be managed and improve the reliability and availability of resources using hybrid fault tolerance scheme.

Keywords: Cloud Computing, Fault Tolerant, Proactive and Reactive Techniques, Self-Healing, VM Failure

1. Introduction

Cloud computing provides the resources based on on-demand, pay-per-use technique through internet. It is an emerging technology, so various challenges that has to be tackled. Among them fault tolerance is the key challenge. Fault tolerance is unavoidable in guaranteeing availability and reliability of cloud applications. To reduce the failure rate of the system, Failures should be predicted and actively handled. In order to achieve an efficient cloud system, fault tolerance techniques are used to predict these failures and proceed with the goal of minimizing the impact of failure. There are two kinds of fault tolerant techniques viz. proactive fault tolerance and reactive fault tolerance. Proactive fault tolerance techniques predict the failure before they occur and prevent them. Reactive fault tolerant techniques reduce the effect of failure on the system after the failure occurs. It simply means to remove the fault from the system. A purely proactive or reactive fault tolerant mechanism will not be effective in providing fault tolerance. Therefore a system involving both proactive and reactive fault tolerance techniques will always be effective in handling failures in the Cloud environment. This work can be implemented using cloudsim simulator. Figure1 shows the overall architecture of cloudsim.

In the work proposed by[1] an approach called Dynamic Proactive Fault Tolerant Technique which decides proactively that on which VM, which applications are to be run based on the capacity of each VM[1]. In[2] have proposed various Fault tolerant techniques based on the policies like Checkpoint recovery Replay and Retry Preemptive migration, Software Rejuvenation and so on[2]. In[3] authors have explained the core simulation framework of CloudSim which is tool for simulating virtualized Cloud-based environments which incorporates the interfaces like VMs, memory, storage, bandwidth etc[3]. The work

*Author for correspondence

published by[4] explains different types of hardware and software fault tolerant techniques including self-healing, replication and so on[4]. In[5] authors have explained in their work about different types of faults that can occur in the cloud environment and compares the existing fault tolerant techniques[5]. According to[6] faults can be injected into the code itself by modifying the SimEntity class in the CloudSim to simulate failure in the cloud environment[6]. In the work proposed by[7] uses improved C4.5 decision algorithm which is an improved ID3 algorithm for detecting fault in the system[7]. The proposed work by[8] explains about when and where to apply proactive and reactive fault tolerant techniques to form an efficient fault tolerant system[8]. In[9] discussed and compared various load balancing techniques in cloud environment[9]. In[10] authors suggested many fault tolerance mechanisms used to tolerate the fault in cloud environment to improve reliability.

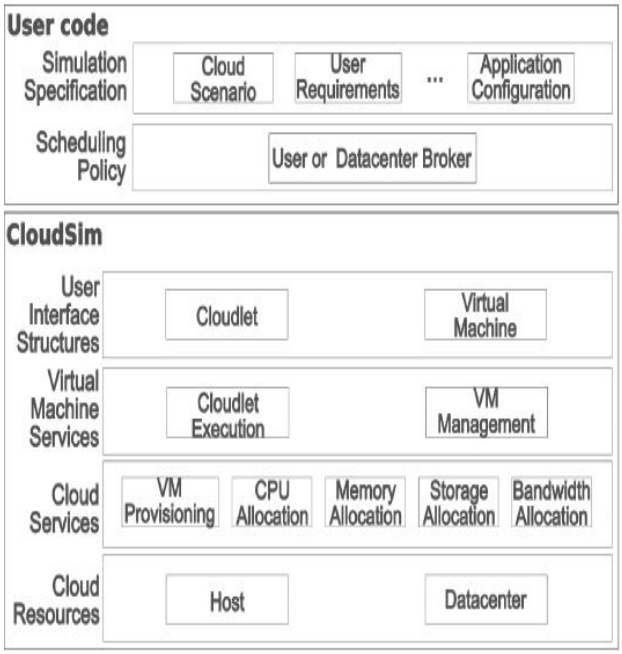

Figure 1. CloudSim architecture.

3. Hybrid Fault Tolerance Scheme

Reliability is ensured through fault tolerance which in turn includes two techniques namely the proactive and reactive fault tolerance. By including these techniques into the Cloud system, failure is either prevented by prediction or damage control is done after the failure occurs. 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 the 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 will yield a Cloud system with increased reliability.

To avoid VM failure due to memory insufficiency or variable capacity change dynamic load balancing in the VM can be done before application 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. The overall flow graph of proposed method illustrates in Figure 2.

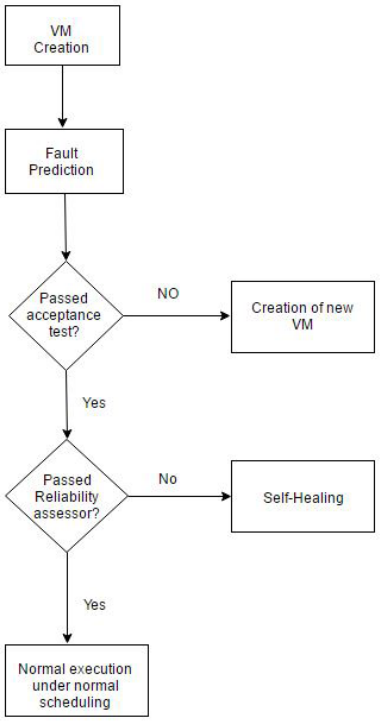

Figure 2. Flow graph of hybrid fault torelance scheme.

The proposed scheme consists of three main steps which are,

- VM creation.
- Fault Prediction.
- Fault Tolerance.

2.1 VM Creation

The creation of VMs under several hosts is done here, using the cloudsim package. The hosts are placed under a data centre and each VM in turn contains several cloudlets under it. The cloudlets are nothing but the applications that run under the VM.

2.2 Fault Prediction

The scheme proposes proactive fault tolerance, so fault has to be predicted before it occurs. Here two tests are used for prediction namely,

Acceptance test

This method checks two conditions. First, the number of processing elements in the VM should be greater than the number of processing elements needed by all the cloudlets under it (VM). Second, the storage size of the VM should be greater than storage size needed by the cloudlets under it. If these two conditions are not satisfied then the VM has a higher chance of failure in the future.

Reliability assessor

This method checks the reliability of the VM by calculating the ability of the VM using the previous history data (average of the data is taken). If the ability is less than the assigned threshold, the reliability of the VM is less and it has a higher chance of failure in the future.

2.3. Hybrid Fault tolerance

Failure in the acceptance test

To handle failure in the acceptance test, a new VM is created under the same host dynamically. The cloudlets under the VM that failed the acceptance test are transferred dynamically to the newly created VM. This newly created VM has the capacity i.e., memory and storage space that is sufficient to execute the cloudlets. Thus the cloudlets can execute under the newly created VM.

Failure in the reliability assessor

To handle failure in the reliability assessor, Self-healing mechanism is performed. In this mechanism, the cloudlets under the VM that failed the reliability assessor are distributed to other VMs under the same host considering the capacity of each VM. Thus the cloudlets are scheduled and executed under the existing VMs.

3. Experimental Result

The proposed method is implemented using cloudsim. It is simulating framework for creating cloud infrastructure. VM creation is the first module of this proposed method. In this implementation, multiple VM are created and all are assigned with different cloudlets. Cloudlet is the task run in the VM. Figure 3 and Figure 4 represent VM creations and Cloudlet creations.

```
Starting program...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker_0 is starting...
Entities started.
0.0: Broker_0: Cloud Resource List received with 1 resource(s)
0.0: Broker_0: Trying to Create VM #0 in Datacenter_0
0.0: Broker_0: Trying to Create VM #1 in Datacenter_0
0.0: Broker_0: Trying to Create VM #2 in Datacenter_0
0.0: Broker_0: Trying to Create VM #3 in Datacenter_0
0.0: Broker_0: Trying to Create VM #4 in Datacenter_0
0.0: Broker_0: Trying to Create VM #5 in Datacenter_0
0.0: Broker_0: Trying to Create VM #6 in Datacenter_0
0.0: Broker_0: Trying to Create VM #7 in Datacenter_0
0.0: Broker_0: Trying to Create VM #9 in Datacenter_0
0.0: Broker_0: Trying to Create VM #8 in Datacenter_0
0.1: Broker_0: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #2 has been created in Datacenter #2, Host #1
0.1: Broker_0: VM #3 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #4 has been created in Datacenter #2, Host #1
0.1: Broker_0: VM #5 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #6 has been created in Datacenter #2, Host #1
0.1: Broker_0: VM #7 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #9 has been created in Datacenter #2, Host #1
0.1: Broker_0: VM #8 has been created in Datacenter #2, Host #0
0.1: Broker_0: Sending cloudlet 0 to VM #0
0.1: Broker_0: Sending cloudlet 1 to VM #1
0.1: Broker_0: Sending cloudlet 2 to VM #2
0.1: Broker_0: Sending cloudlet 3 to VM #3
0.1: Broker_0: Sending cloudlet 4 to VM #4
0.1: Broker_0: Sending cloudlet 5 to VM #5
0.1: Broker_0: Sending cloudlet 6 to VM #6
0.1: Broker_0: Sending cloudlet 7 to VM #7
0.1: Broker_0: Sending cloudlet 8 to VM #9
0.1: Broker_0: Sending cloudlet 9 to VM #8
0.1: Broker_0: Sending cloudlet 10 to VM #0
```

Figure 3. VM creation.

```
0.1: Broker_0: Sending cloudlet 11 to VM #1
0.1: Broker_0: Sending cloudlet 12 to VM #2
0.1: Broker_0: Sending cloudlet 13 to VM #3
0.1: Broker_0: Sending cloudlet 14 to VM #4
0.1: Broker_0: Sending cloudlet 15 to VM #5
0.1: Broker_0: Sending cloudlet 16 to VM #6
0.1: Broker_0: Sending cloudlet 17 to VM #7
0.1: Broker_0: Sending cloudlet 18 to VM #9
0.1: Broker_0: Sending cloudlet 19 to VM #8


200.0: The simulation is paused for 1 sec


200.0: workload failed in acceptance test for the vm # 8

320.1: Broker_0: Cloudlet 0 received by 0
320.1: Broker_0: Cloudlet 10 received by 0
320.1: Broker_0: Cloudlet 1 received by 1
320.1: Broker_0: Cloudlet 11 received by 1
320.1: Broker_0: Cloudlet 3 received by 3
320.1: Broker_0: Cloudlet 13 received by 3
320.1: Broker_0: Cloudlet 5 received by 5
320.1: Broker_0: Cloudlet 15 received by 5
320.1: Broker_0: Cloudlet 7 received by 7
320.1: Broker_0: Cloudlet 17 received by 7
320.1: Broker_0: Cloudlet 2 received by 2
320.1: Broker_0: Cloudlet 12 received by 2
320.1: Broker_0: Cloudlet 4 received by 4
320.1: Broker_0: Cloudlet 14 received by 4
320.1: Broker_0: Cloudlet 6 received by 6
320.1: Broker_0: Cloudlet 16 received by 6
320.1: Broker_0: Cloudlet 8 received by 9
320.1: Broker_0: Cloudlet 18 received by 9
```

Figure 4. Cloudlet creations.

In existing schemes, fault tolerance can be achieved when VM's are static mode. In proposed method, VM's faults can be predicted and tolerance can be achieved while VM's are in running state. Figure 5 illustrates how

fault prediction and tolerance can be done. Figure 6 describes clearly the final result after tolerating the VM faults and manage its performance. Tolerance is transparent to the user.

```
960.1: Broker_0: Cloudlet 0 received by 0
960.1: Broker_0: Cloudlet 10 received by 0
960.1: Broker_0: Cloudlet 1 received by 1
960.1: Broker_0: Cloudlet 11 received by 1
960.1: Broker_0: Cloudlet 2 received by 2
960.1: Broker_0: Cloudlet 12 received by 2
960.1: Broker_0: Cloudlet 3 received by 3
960.1: Broker_0: Cloudlet 13 received by 3
960.1: Broker_0: Cloudlet 5 received by 5
960.1: Broker_0: Cloudlet 15 received by 5
960.1: Broker_0: Cloudlet 7 received by 7
960.1: Broker_0: Cloudlet 17 received by 7
960.1: Broker_0: Cloudlet 9 received by 10
960.1: Broker_0: Cloudlet 19 received by 10
960.1: Broker_0: Cloudlet 4 received by 4
960.1: Broker_0: Cloudlet 14 received by 4
960.1: Broker_0: Cloudlet 6 received by 6
960.1: Broker_0: Cloudlet 16 received by 6
960.1: Broker_0: Cloudlet 8 received by 9
960.1: Broker_0: Cloudlet 18 received by 9
960.1: Broker_0: All Cloudlets executed. Finishing...
960.1: Broker_0: Destroying VM #0
960.1: Broker_0: Destroying VM #1
960.1: Broker_0: Destroying VM #2
960.1: Broker_0: Destroying VM #3
960.1: Broker_0: Destroying VM #4
960.1: Broker_0: Destroying VM #5
960.1: Broker_0: Destroying VM #6
960.1: Broker_0: Destroying VM #7
960.1: Broker_0: Destroying VM #9
960.1: Broker_0: Destroying VM #10
Broker_0 is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
```

Figure 5. Fault prediction and tolerance.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID       Time    Start Time    Finish Time
    0          SUCCESS        2             0          320       640.1          960.1
    1          SUCCESS        2             1          320       640.1          960.1
    2          SUCCESS        2             2          320       640.1          960.1
    3          SUCCESS        2             3          320       640.1          960.1
    4          SUCCESS        2             4          320       640.1          960.1
    5          SUCCESS        2             5          320       640.1          960.1
    6          SUCCESS        2             6          320       640.1          960.1
    7          SUCCESS        2             7          320       640.1          960.1
    8          SUCCESS        2             9          320       640.1          960.1
    9          SUCCESS        2             10          320       640.1          960.1
    10          SUCCESS        2             0          320       640.1          960.1
    11          SUCCESS        2             1          320       640.1          960.1
    12          SUCCESS        2             2          320       640.1          960.1
    13          SUCCESS        2             3          320       640.1          960.1
    14          SUCCESS        2             4          320       640.1          960.1
    15          SUCCESS        2             5          320       640.1          960.1
    16          SUCCESS        2             6          320       640.1          960.1
    17          SUCCESS        2             7          320       640.1          960.1
    18          SUCCESS        2             9          320       640.1          960.1
    19          SUCCESS        2             10          320       640.1          960.1


1060.1: Vm #8 failed
```

Figure 6. Final result after fault tolerance.

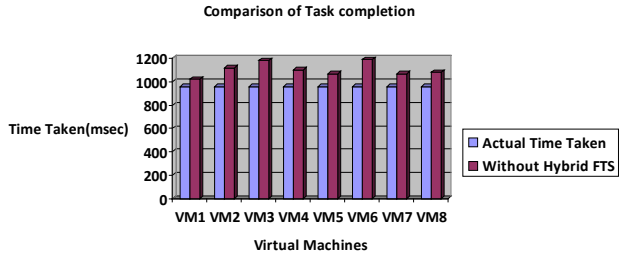

Figure 7. Comparison of task completion without using hybrid Fault Tolerance Scheme(FTS).

The Figure 7 represents the relationship between actual time taken by VM to complete without fault occurrence and with fault occurrence. It illustrates that time taken to complete the cloudlet task with fault occurrence decreases the performance of the cloud simulator environment.

The Figure 8 shows that if proposed hybrid Fault Tolerance Scheme (FTS) is used for tolerating the faults, each VM can completes its cloudlets task within actual time taken. Proposed scheme tolerates the fault while VM is in running state with transparency.

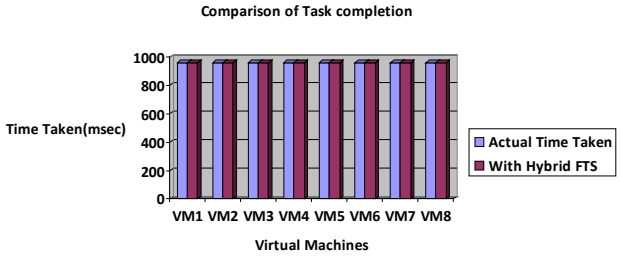

Figure 8. Comparison of task completion with using hybrid Fault Tolerance Scheme(FTS).

4. Conclusion and Future Work

In proposed paper, VM failure is predicted and handled gracefully while the applications in the VM have started run to already. The fault tolerant technique proposed is a combination of both proactive and reactive techniques making this a hybrid fault tolerant system. For implementing the proposed work cloudsim simulation tool has been used. In future, this work can be applied in a real cloud environment and checked for its efficiency. Further improvement can be done by including the live migration technique into the system. The proposed work on predicting the faulty VM, transfers the cloudlets under it to other VMs or a new VM and they are executed again. This can be improved by using the live migration technique to transfer the cloudlets to other VMs.

5. References

1. Goutam D, Verma A, Agrawa N. The performance evaluation of proactive fault tolerant scheme over cloud using CloudSim Simulator. 5th International Conference on the Application of Digital Information and Web Technologies; 2014.

2. Bala A, Chana I. Fault tolerance- challenges, techniques and implementation in cloud computing. IJCSI International Journal of Computer Science Issues. 2012 Jan; 9(1):288–93.
3. Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, software- practice and experience. 2011; 41:23–50.
4. Kaur J, Kinger S. Analysis of different techniques used for fault tolerance. International Journal of Computer Science and Information Technologies. 2014; 5(3):4086–90.
5. Kaur H, Kaur A. A survey on fault tolerance techniques in cloud computing. International Journal of Scientific Engineering and Applied Science. 2015; 1(9):419–25.
6. Nita MC, Pop F, Mocanu M, Cristea V. FIM-SIM: Fault injection module for CloudSim based on statistical distributions. Journal of Telecommunication and Information Technology. 2014; 4:14–23.
7. Jiang Y, Huang J, Ding J, Liu Y. Method of fault detection in cloud computing systems. International Journal of Grid Distribution Computing. 2014; 7(3):205–12.
8. Ganesh A, Sandhya M, Shankar S. A study on fault tolerance methods in cloud computing. IEEE International Advance Computing Conference. 2014.
9. Verma A, Sharma SK, Kaur B. Comparative analysis on load balancing techniques in cloud computing. Indian Journal of Science and Technology. 2016; 9(11):1–11.
10. Padmakumari P, Umamakeswari A. Methodical review on various fault tolerant and monitoring mechanisms to improve reliability on cloud environment. Indian Journal of Science and Technology. 2015; 8(35):1–6.