JES
Journal of Engineering Sciences
Vol 11, Issue 4, APRIL /2020
ISSN NO:0377-9254

Quality of Prediction in Fault Tolerance Management System on Virtual Machines

Vaddhiraju Swathi,
Assistant Professor,Dept of CSE,
swathikasarla1980@gmail.com
Sreyas institute of engineering and Technology, Hyderabad.

***Abstract*-Cloud computing is increasingly attracting huge attention both in academic research and industry initiatives and has been widely used to solve advanced computation problem. the Design and Performance analysis of a Cloud Based Virtual Machine Success and Failure Rate in a typical Cloud Computing Environment and Prediction Methods. a new reliable Fault Tolerance scheme using an intelligent optimal strategy is presented to ensure high system availability, reduced task completion time and efficient VM allocation process. Specifically, (i) A generic fault tolerance algorithm for cloud datacentres and high-performance computing(HPC) systems in the cloud was developed. (ii) A verification process is developed to a fully dimensional VM specification during allocation in the presence of fault. (iii) A failure prediction model is further developed, and the predictive capabilities of machine learning is explored by applying several algorithms to improve the quality of prediction.**

Keywords: Fault tolerance, Cloudsim, Virtual Machine, Fault Injector,QoS.

1.Introduction

Cloud computing is increasingly attracting huge attention both in academic research and in industrial initiatives. It has become a popular paradigm and an attractive model for providing computing, IT infrastructure, network and storage to end users in both large and small business enterprises [1]. The surge in cloud popularity is mainly driven by its promise of an on-demand flexibility and scalability, without committing any upfront investment in implementation, with reduction in operating costs of infrastructure and datacentres [2]. cloudbased high-performance systems, big-data and social networking, the demand for additional computational power and resources supported by cloud computing platforms has increased, thereby increasing the probability and risk of failure[3]. Hence, the need for a reliable fault tolerant management system is very vital to efficiently ensure the smooth running and operations of a datacentre infrastructure. In order to achieve and maintain a high level of reliability and availability, fault tolerance (FT) becomes an important property that must be considered because it allows the system to continue functioning correctly in the event of failure. Embedding a fault-tolerant design in a cloud architecture allows the system to continue its intended operation

even at areduced level of performance rather than breaking down completely in the event of an unexpected failure [4, 5].

2. Aims and Objectives

The aim of this work is to develop an effective fault tolerance management system on the virtual machines by providing high availability in a cloud environment using a proactive optimized approach, checkpoint mechanism, machine learning and a robust fault tolerance approach for cloud data centers ensuring acceptable quality of service (QoS). The main objectives are as follows:

- To develop an efficient smart fault tolerance approach for cloud datacentres, using the pass rate of computing virtual nodes, decision mechanism algorithm and a fault manager by applying an optimized checkpoint strategy in a virtualized environment.
- To provide reliable fault tolerance scheme by developing a framework, implementing the associated algorithms and testing the proposed solution through simulations.
- To test, validate and compare the performance of the proposed fault tolerance management strategy with the default existing algorithm using CloudSim simulation toolkit in the presence fault injection module.
- To conduct a performance comparison of existing approaches with the proposed strategy and to validate through experimental results and mathematical models using Cloud simulation toolkit as a cloud simulator.
- To develop an effective failure prediction model using time series and machine learning utilising real data from a HPC data center infrastructure, and evaluating the experimental results through support vector machine (SVM), random forest (RF), k-nearest neighbours(k-NN), linear discriminant analysis (LDA) and classification and regression trees (CART).

3.CloudSim Simulation Framework

the objectives of CloudSim project and the benefits by the simulation-based approaches to researchers as against the test-bed platform, the following are some key benefits: (i) Simulation-based platform helps to test application services in repeatable and controllable environment. (ii) They help to tune and adjust the system bottlenecks before deploying application on the actual cloud.

Vol 11, Issue 4, APRIL /2020
ISSN NO:0377-9254

(iii) The platforms help to experiment and verify the services on simulated infrastructures with varying workload mix and different resource performance scenarios to experiment and develop adaptive application provisioning policies and techniques. (iv) Simulation experiment cost much lesser compared to cost of purchasing and setting up a test-bed for experiment.

3.1.Features of CloudSim

- It supports the modelling and simulation of a large-scale Cloud computing datacentre environment.
- It enables the simulation and modelling of a virtualised server hosts, with customizable policies for provisioning host resources to virtual machines.
- It supports the simulation and modelling of energy-aware computational resources, datacentre network topologies, application containers and message-passing applications
- It offers an autonomous platform for dynamic insertion of simulation elements, stop and resume of simulation, modelling and simulation of federated clouds.
- It also enables support for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines.

4. CloudSim classes design and component characteristics

The CloudSim toolkit [6] classes architecture and characteristics are presented as well as the general features. This is because understanding the Cloud computing environment structure, its architecture and working component and how the inter-object of cloud such as datacentre, broker and host communicates with each other is very vital [7]. To understand the behaviour and analyse the scheduling, allocation, load balancing and fault tolerance algorithms, knowing the classes architecture and component interaction is very useful to the overall design. In addition, understanding of how the CloudSim toolkit is modelled and implemented in the cloud environment makes the design a bit easy and straight forward. The breakdown of the classes, how the inter-objects are modelled and implemented, as well as the architecture of the toolkit environment has the following features as presented in Figure 1 below. The CloudSim toolkit design classes source code can be

found in[8]. The CloudSim system or environment setup is made up of the following three key components which are the broker component, cloud side component and the host component.

5. The Algorithm testing, selection criteria

5.1.The Testing Phase:

This phase defines the rules on each host and decides either the host has the ability to create the M or not. Each Host is checked if it has the ability to carry out the provisioning of the resources for the VM, then returns the result to the Datacenter Broker as a Boolean value (True/False)[9].

5.2. The Fault Injection and Management Phase:

This phase injects some form of fault in the system using the fault injection module. The fault could be in form of excess Load introduction, VM failure (making some VM fail by purposely switching them off) or introducing noise and observing how it affects the system performance, failure rate, success rate and response time[10].

5.3. The Selection and Decision-making Phase:

This phase defines the rules of how the broker selects a suitable host or a successful VM with the highest number of Pass rate. The Datacenter Broker updates all results (true/false) in CIS table, then in ascending order starts attempting to allocate and create the VM just for a suitable host who has a true value. The context of this work aims to verify the concept of testing the default VM allocation algorithm by injecting fault using the fault injection module and comparing it with the proposed FT algorithm and observing the response time and performance. The next section will introduce the default virtual machine allocation policy and the design and implementation of the VM FT Policy and its algorithm through the CloudSim toolkit.

5.4. The Default Virtual Node Provisioning Policy

Algorithm 1 below shows the steps of the default policy algorithm in Cloud Sim. The experiment was initiated by running the default virtual node2 provisioning policy to understand the default process and steps taken place in the algorithm before a virtual node is allocated or provisioned.The algorithm is quite straight forward and based on a few simple steps, which depends on one dimension and VM specification in order to identify, allocate and provision the proper host according to its instructions.

Consequently, it is considered as one-dimensional VM provisioning policy because it depends on one VM specification during the provisioning and allocation process, which is the number of PEs. Based on the algorithm, the policy tries to create the Virtual Node3 instance directly after finding and identifying the proper host. Therefore, if the creation process is successfully done, then the policy returns a True result and if it fails then the policy returns a False result.

Algorithm 1: Default Virtual Node Allocation Policy

Variables*: VirtualNodeTable Map, AvailablePEs List, UsedPEs List,*

Input*: **VN** Object*

Output*: Boolean **Result, (True/False)***

1. ***Result****=False/ True/ Tries=0,1;*
2. **If** (***Virtual Node*** is not created)
3. **Do:**
4. **For All** (host available in *Host.List*)
5. **If** (Primary*Host.Pes >= needed.PEs*)
6. *idx*= index of the main host*;*
7. Break
8. **End If**
9. **End For**
10. *Main Host = PrimaryHostList.get_Host*(*idx*);
11. ***Result*** = Create_VirtualNode(*Host*);
12. *Tries++;*
13. **If (*Result*)**
14. *Update:*
15. *VirtualNodeTable* ***(VNT)***
16. *Used_PEs (needed.Pes)*
17. *Available_PEs (needed. Pes)*
18. ***Result*** **=** true
19. Break
20. **Else**
21. set the Primary*Host.Pes* in minimum value;
22. **End If**
23. **While** (!***Result*** && Tries < *Available_PEs.size*);
24. **End If**
25. Return ***Result***

6.Validation of the proposed VM Fault Tolerance policy

Having introduced the default VM allocation algorithm,tries to evaluate the proposed VM Fault tolerance algorithm usingpractical numerical experiments performed through the CloudSim toolkit. To further validate the concept of the proposed algorithm, some experiments were performed to introduce a Fault Injection Module and then observe the VM distribution, failure rate, success rate, response time and performance over two key scenarios. They are the Time-Shared VM Scheduler scenario and Space-Shared VM Scheduler scenario[11], [12]. These two VM schedulers scenarios are the default ones available in the CloudSim toolkit [13].

6.1.Fault Injection Module

An overview of the fault injection module[10], its algorithm and the integration with CloudSim simulation toolkit. The Fault tolerance fault injection module was

extended from the CloudSim core functions and is made up of three entities as illustrated.

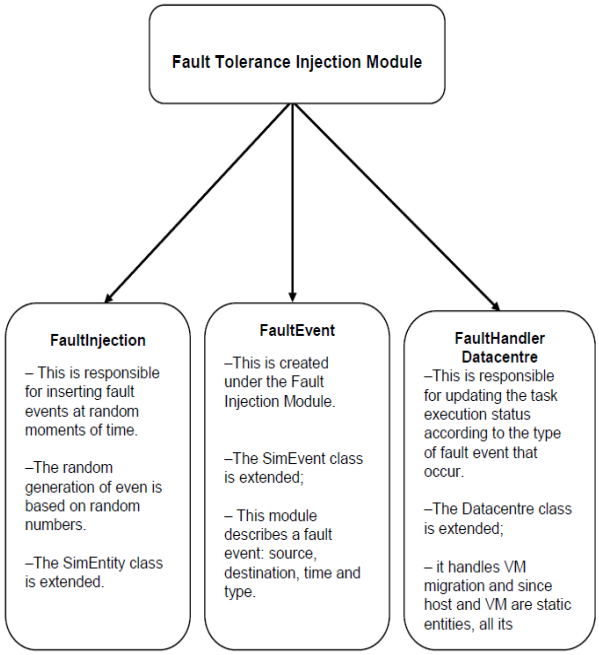

Figure 1: Fault Injection Module[10]

Algorithm 2 below presents the steps involved in the proposed fault injection module algorithm, where fault was introduced into the system and some metrics were observed.

Algorithm 2: Fault Injector Module Algorithm

Variables: *VirtualNodeTable Map, AvailablePEs List, UsedPEs List,*

Input: ***VN*** *Object*

Output: *Boolean* ***Result True/False***

1. ***Result****=False/ True/ Tries=0,1;*
2. **If** (***Virtual Node*** is not created)
3. **Do:**
4. **For All** (host available in *Host.List*)
5. **If** (Primary*Host.Pes >= needed.Pes*)
6. *idx*= index of the primary host*;*
7. Break
8. *sendNow(dataCenter.getId(),*
9. FaultEventTags.VM FAILURE,
10. host);
11. *Mean = RandomNumbers.mean*;
12. *X = RandomNumbers.sample();*
13. **If** *(x > mean)*
14. | GererateFault();
15. } **Else** {
16. Continue;
17 }
17. **End if**
18. **End For**
19. *Main Host = PrimaryHostList.get_Host*(*idx*);
20. ***Result*** = Create_VirtualNode(*Host*);
21. *Tries++;*
22. **If (*Result*)**
23. *Update:*
24. *VirtualNodeTable* (***VNT***)
25. *Used_PEs (needed.PEs)*
26. *Available_PEs (needed. PEs)*
27. ***Result*** = true
28. Break
29. **Else**

30. set the Primary*Host.PEs* in minimum value;

31. End If

32. While (!***Result*** && Tries < *Available_PEs.size*);

33. 34 **End If**

34. Return ***Result***

The fault injection module is one of the important components of this research because it helps to further validate the concept of the proposed algorithm by testing the effect of introducing fault into the system and how robust the proposed algorithm will perform in terms of withstanding fault and during failure scenarios. As earlier stated, some metrics are observed under fault injection scenarios, and this is compared with the default algorithm without the proposed algorithm, then a performance comparison is performed to evaluate scheme.

7.Conclusion

A fault tolerance framework for cloud datacentres and HPC systems that can be used to build a reliable and efficient system by providing high availability depending on cloud user requests to successful virtual machines (VMs) using the proposed intelligent FT algorithm. Developed an integrated virtualised failover strategy for cloud data centers, overall reducing the system service time, VM allocation time and improving the overall system performance. We validate our algorithm by injecting fault and comparing it with the default CloudSim algorithm.

8.References

1) K. Bilal, O. Khalid, S. U. . Malik, M. U. S. Khan, S. . Khan, and A. . Zomaya, "Fault Tolerance in the Cloud," *"Fault Tolerance in the Cloud" Encyclopedia on Cloud Computing*. John Wiley & Sons, Hoboken, NJ, USA, 2015, pp. 291–300, 2015.

2) O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Cloud computing migration and IT resources rationalization," *2014 Int. Conf. Multimed. Comput. Syst.*, pp. 1164–1168, Apr. 2014.

3) Gainaru and F. Cappello, *Fault-Tolerance Techniques for High-Performance Computing*. 2015.

4) S. Patidar, D. Rane, and P. Jain, "A survey paper on cloud computing," *Proc. - 2012 2nd Int. Conf. Adv. Comput. Commun. Technol. ACCT 2012*, pp. 394–398, 2011.

5) Tchana, L. Broto, and D. Hagimont, "Approaches to cloud computing fault tolerance," *IEEE CITS 2012 - 2012 Int.*

Conf. Comput. Inf.Telecommun. Syst., 2012.

6) T. C. C. and D. S. (CLOUDS) Laboratory, "The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne-DatacenterBroker (cloudsim 3.0 API)." [Online]. Available: [Accessed: 26-Sep-2018].

7) T. C. C. and D. S. (CLOUDS) Laboratory, "The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne-VmAllocationPolicySimple (cloudsim 3.0 API)." [Online]. Available: ocationPolicySimple.html. [Accessed: 26-Sep-2018].

8) Cloudsim.org, "The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne," *CloudSim Project Google Group*, 2013. [Online]. Available: http://www.cloudbus.org/cloudsim/. [Accessed: 10-Oct-2018].

9) T. C. C. and D. S. (CLOUDS) Laboratory, "The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne-DatacenterBroker (cloudsim 3.0 API)." [Online]. Available: [Accessed: 26-Sep-2018].

10) M. Nita, F. Pop, M. Mocanu, and V. Cristea, "FIM-SIM : Fault Injection Module for CloudSim Based on Statistical Distributions," *J. Telecommun. Inf. Technol.*, pp. 14–23, 2014.

11) T. C. C. and D. S. (CLOUDS) Laboratory, "The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne-VmSchedulerTimeShared (cloudsim 3.0 API)." [Online]. Available: [Accessed: 26-Sep-2018].

12) T. C. C. and D. S. (CLOUDS) Laboratory, "The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne-VmSchedulerSpaceShared (cloudsim 3.0 API)." [Online]. Available: [Accessed: 26-Sep-2018].

13) T. C. C. and D. S. (CLOUDS) Laboratory, "The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne-VmScheduler (cloudsim 3.0 API)." [Online]. Available: [Accessed: 26-Sep-2018].