ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st& 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Improving Fault Tolerance in Virtual Machine Based Cloud Infrastructure

Rajesh.S, Kanniga Devi.R

Graduate student, Dept of Computer Science and Engineering, Kalasalingam University, Tamil Nadu, India.

Dept of Computer Science and Engineering, Kalasalingam University, Tamil Nadu, India,

Abstract— Cloud Computing is a style of Computing where service is provided across the internet using different models. Fault tolerance is a major concern to guarantee availability and reliability of critical services as well as application execution. In this project work, we propose a model to analyze how system tolerates the faults and make decision on the basics of reliability of the processing nodes, i.e. Virtual machines. If a virtual machine manages to produce a correct result within the time limit, its reliability increases, and if it fails to produce the result within time or correct result, its reliability decreases. If the node continues to fail, it is removed, and a new node is added. There is also a minimum reliability level. If any processing node does not achieve that level, the system will perform backward recovery or safety measures. The proposed technique is based on the execution of design diverse variants on multiple virtual machines, and assigning reliability to the results produced by variants. The virtual machine instances can be of same type or of different types. The system provides both the forward and backward recovery mechanism, but main focus is on forward recovery.

Keywords— cloud computing, virtual machine, fault tolerance, reliability

I.INTRODUCTION

Cloud computing share the resources like physical services, storage, and networking. The cloud offers many services through cloud service providers. The most popular cloud service providers are Google, Amazon, windows Azure etc...Each service provider provides different services based on the demand of the users. For example Amazon provide IaaS service Google Provides all services like SaaS, PaaS and IaaS. The cloud computing is based on the distributed concepts and it is reliable to all users. This paper deals with the research field of fault tolerance in cloud. In a cloud environment there are many unknown nodes called Virtual machines (VM). Virtual machine (VM) is an operating system (OS) or program that can be installed and run virtually. In other words, VM is a processing machine in the server. In cloud computing, the user data is replicated in many VM's.

The user request is passed onto all the available VM's. If the particular VM fails then it will not respond and, all the other active VMs respond to the request. The fault tolerance measure available should identify the one reliable VM among all the VM's and respond to the client request. This paper is used to identify the reliable VM.

II.FAULT TOLEANCE AS A RESEARCH ISSUE

Fault tolerance is a major problem to guarantee availability and reliability of critical services as well as application execution. Fault tolerance serve as an effective means to address reliability concerns. Fault tolerance means that system should continue to operate under fault presence.

Cloud is vulnerable to a large number of system failures and the traditional fault tolerance approaches are less effective since cloud system's architectural details are not widely available to the users because of the abstraction layers and business model of cloud computing. Fault tolerance that uses the Virtualization Technology (VT) can increase the reliability of applications, but VM migration and consolidations are difficult to achieve.

There are various faults which can occur in cloud computing .Based on fault tolerance policies various fault tolerance techniques[1] can be used that can either be task level or workflow level .

REACTIVE FAULT TOLERANCE: Reactive [1] fault tolerance policies reduce the effect of Failures on application execution when the failure effectively occurs. There are various techniques which are based on these policies like Checkpoint/Restart, Replay and Retry and so on.

PROACTIVE FAULT TOLERANCE: The principle of proactive [1] fault tolerance policies is to avoid recovery from faults, errors and failures by predicting them and proactively replace the suspected components by other working components. Some of the techniques which are based on these policies are Preemptive Migration, Software Rejuvenation etc.

III.RELATED WORK

A lot of work has been done in the area of fault tolerance for standard cloud infrastructure. But there is lot of research room available in fault tolerance of virtual machine (VM) based cloud infrastructure. Cloud infrastructure has introduced some new issues related to Fault tolerance. These characteristics are different from the existing traditional techniques.

Huang et.al., [2] present Algorithm-Based Fault Tolerance (ABFT) method. ABFT uses matrix or vector level checksum in row and column to detect a faulty processor in multiple processor systems. The method can be used to detect and correct errors in matrix operations such as addition, multiplication, scalar product, and LU-decomposition performed in multiple processor systems which may have one failed processor. In their work, they focus on the problem of achieving a certain reliability with the minimum cost in potentially faulty clouds.

Wenbing Zhao et.al., [12] propose a FT middleware which implement a synchronized server replication strategy, where a failed server is repaired with a consistent state.

Alain Tchana et.al., [13] suggest a fault tolerance method collaborating cloud provider and cloud customer. Their integrated approach makes fault tolerance available in all levels of the cloud. However, they are not making use of VM checkpoint solutions to achieve optimum fault tolerance.

Z. Dai et.al., [3] suggest transparent check pointing at the user's level provided by Distributed Multithreaded Check Pointing. By considering economic and dependability factors check points with various parameters are fixed. If these parameters are not satisfied, the thread is restarted.

X. Kong et.al., [5] presents a model for virtual infrastructure performance and fault tolerance. But it is not well suited for the fault tolerance of real time cloud applications. For the non-cloud applications, a baseline model for distributed RTS is, distributed recovery block proposed by K. H. Kim which is very basic in nature.

J. Coenen et.al., [6] propose model, "A formal approach for the fault tolerance of distributed real time system (RTS)" Traditionally, one of the backbones of software reliability is avoiding the faults. Since cloud architecture is very complex and built on data center comprising thousands of interconnected servers with capability of hosting a large number of applications and distributed globally, fault prevention techniques in developing stage is very tedious. Fault avoidance techniques or fault removal techniques such as testing to detect and remove fault, therefore, won't be enough in the case of cloud computing.

S. Malik et.al., [18] propose model "Time stamped fault tolerance of distributed RTS". This model incorporated the concept of time stamping with the outputs. All of these models were defined for the real time systems based on standard computing architecture.

Slawinska et.al., [14] proposes that in order to overcome VM failure, a Virtual Machine hook can be set to "resubmit" the failed VM. VM Crash is recovered by "one VM restart" functionality. Windows Azure offers Fault Tolerance management with the replicas of each VM and this solution is limited to the applications developed in the Windows Azure platform. VM failure of Amazon EC2 is take care by Simple Queue Service (SQS) and Amazon Machine Image (AMI). Service requests are queued up till they are executed properly, or deleted by the user with the help of SQS. In EC2 we can publish many Amazon Machine Images (AMI), on the failure of an AMI, we can easily replace it with the help of an API invocation.

H.Chen et.al., [15] proposes that the function of fault tolerance is to preserve the delivery of expected services despite the presence of fault-caused errors within the system itself. Errors are detected and corrected, and permanent faults are located and removed while the system continues to deliver acceptable service. This goal is accomplished by the use of error detection algorithms, fault diagnosis, recovery algorithms and spare resources.

Antonina Litvinova et.al., [7] use active replication techniques for web services, and propose a technique to gain byzantine fault tolerance using virtualization technology. Techniques to build efficient and fault tolerant applications for Amazon's EC2 are provided in. Another approach using fault tolerance middleware which follows a leader/follower replication approach to tolerate crash faults has been proposed in. However, all these techniques have the limitations and either tolerate only a specific kind of fault or provide a single method to resilience.

IV.PROPOSED WORK

Our proposed model is based upon improving reliability assessment of virtual machines in cloud environment and fault tolerance of applications running on those VM's. This fault tolerance has to be done on the basis of the reliability of virtual machines. The proposed method use two different set of nodes. One is the set of VM and other is adjudication node such as the main node (server). The Virtual machine uses acceptance test for its

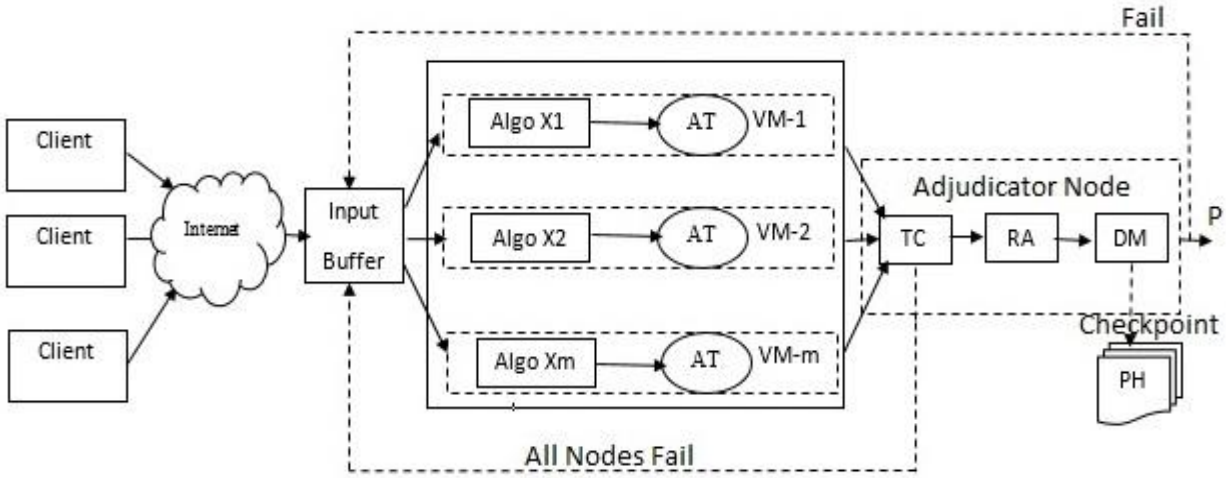

Figure-1: Proposed System Model

logical validity. The adjudicator node contains the time checker, reliability assessor and decision mechanism algorithms to find the reliable VM. The reliable VM is identified to process the client request. The client can accept the data from the VM in compressed form. A virtual machine is selected for computation on basis of maximum and minimum reliability. The node with maximum reliability is selected as the system event output. To provide the fault tolerance the data can be stored on multiple cloud using virtualization techniques.

ACCEPTANCE TEST (AT): Acceptance Test (AT) module checks whether the fault will occur or not. Here the fault is a failure of VM or Host. The acceptance test can respond both success and failure case of the VM. If the VM has failed that VM is not considered. Even when the data within a VM gets corrupted it will not affect VM function so this VM considered by the TC. If result is success then it passes result to TC module. If the result is failure then it does not pass the result to TC module. To indicate the failure of result, AT sends a verify exception signal to TC.

TIME CHECKER (TC): Time checker is evaluated in milliseconds. The time is given as a lower bound time limit and the upper bound time limit. In our process the response time limit for each VM is given milliseconds. It must be monitored for every milliseconds. The VM can respond within the specified time limit, and that VM is taken as reliable then passed to RA (Reliability Assessor) module .TC module raises the signal of overtime which produce the result in deadline time. If all the nodes fail then TC module performs the recovery.

RELIABILITY ASSESSOR (RA): The RA module assesses the reliability for each virtual machine. The reliability is identified based on the main core module of

the proposed system. As the proposed system tolerates the faults and makes the decision on the basis of the reliability of the processing nodes (i.e. virtual machine), the reliability of the virtual machine is improved, which

changes after every computing cycle. In the beginning the reliability of each virtual machine is 100%. If a processing node manages to produce a correct result within the time limit, its reliability increases using a reliability Factor (RF), and if the processing node fails to produce the correct result or result within the time limit, its reliability decreases using adaptability factor n. The reliability assessment algorithm is more convergent towards failure conditions. In RA, the VM which responds above the time limit, is considered as failure VM.

```
Begin
Initially rel:=1, n :=1
Input from configuration RF, maxRel,
minRel
Input nodestatus
if nodeStatus =Pass then
rel := rel + (rel * RF)
if n > 1 then
n := n-1;
else if nodeStatus = Fail then
reliability := reliability – (rel * RF * n)
n := n+1;
if reliability >= maxRel then
reliability := maxRel
if reliability < minRel then
nodeStatus :=dead
call_proc: remove_this_node
call_proc: add_new_node
End
```

Algorithm-1: Reliability Assessment

DECISION MECHANISM (DM): Identify the Reliable virtual machine based on Resource availability and previous history.

Resource Availability: Memory is considered as resource. The memory availability for each VM is considered separately. We apply the memory availability algorithm to find the lowest memory utilization VM and take that VM as reliable.

Previous History: It is a repository area to hold the checkpoints. At the end of each computing cycle DM makes checkpoint in it. In case of all node failure, backward recovery is performed with the help of checkpoints maintained in this Previous History. In our experiment we have done the communication induced checkpoint (CIC). The CIC perform the check pointing at the end of every cycle to maintain a global state. This scheme provides an automatic forward recovery. If a node fail to produce output or produce output after time overrun the system will not fail. It will continue to operate with remaining nodes. This mechanism will produce output until all the nodes fail.

Begin
Initially rel:=1, n :=1
Input from RA nodeRel, numCandNodes
Input from configuration SRL
bestRel := find_reliability of node with highest reliability
if bestRel >= SRL
status := success
else
perform_backward_recovery
call_proc: remove_node_minRel
call_proc: add_new_node
End

Algorithm-2: Decision Mechanism

The reliability assessment algorithm is executed for each node (virtual machine). Initially reliability of a VM is set to 1. There is an adaptability factor n, which controls the adaptability of reliability assessment. The value of n is always greater than 0. The algorithm takes input of three factors RF (Reliability Factor), minReliability (Minimum Reliability) and maxReliability (Maximum Reliability). RF is a reliability factor which increases or decreases the reliability of the node. It decreases the reliability of the node more quickly as compare to the increase in reliability. It happen due to its multiplication with the adaptability factor n. minReliability is the minimum reliability level. If a node reaches to this minReliability level, it is stopped to move further operations. maxReliability is the maximum reliability level. It is really important in a situation, where a initially produces correct results in consecutive cycles, but then fails again and again. So its reliability should not be high enough to make the reliability difficult to decrease and converge towards lower reliability. The algorithm is normally more convergent to failures in near preceding. So if there are two nodes and both of them have 10 passes and 10 failures in total 20 cycles. But the node, who have more failures in near past has more chances to have lesser reliability than the other. This factor is really in accordance to latency issues, where initially node latency was good, but then it becomes high. So this node tends to more node failures by failing to produce the results in time. The values of variables (RF, minReliability, maxReliability, SRL) depend on the applications.

V.EXPERIMENTAL SETUP

We simulated our experiment using CloudSim. CloudSim [30] is a Cloud computing modeling and simulation tool that was developed in the University of Melbourne, Australia. It aims to provide Cloud computing re-searchers with a comprehensive experimental tool to conduct new research approaches. It supports the modeling and simulation of large scale Cloud computing environments, including power management, performance, data centers, computing nodes, resource provisioning, and virtual machine provisioning. We take 12 Virtual Machine in this experiment. The virtual machines are configured as follows:

TABLE-1: VM CONFIGURATION	
PARAMETER	**VALUE**
Architecture	X86
OS	LINUX
VMM	XEN
Cost	3.0
Costpermem	0.05
Costperstorage	0.001
Costperbw	0.0

TABLE-2: HOST CONFIGURATION	
PARAMETER	**VALUE**
RAM	10000
STORAGE	1000000
BW	100000

TABLE-3: DATACENTER CONFIGURATION	
PARAMETER	**VALUE**
Architecture	X86
OS	Linux
Storage Cost $/s	0.1
Data Transfer Cost $/Gb	0.1
Physical HW units	2

The adjudication node sends data to VM's and receiving the results from the VM's. The response time for the VM is taken from lower limit 1 to upper limit 1.5 milliseconds. All Virtual Machines execute the algorithm simultaneously.

VII.RESULT

A metric analysis is given for the reliability assessment impact analysis. Here we have analyzed the reliability improved for Virtual Machine. We have assumed that the value of reliability factor (RF) is 2% (i.e. 0.02). Initially, the reliability is 1. Comparison is made between host and virtual Machine. This

comparison is done for 15 Cloudlets 12 Virtual Machine and 12 Host.

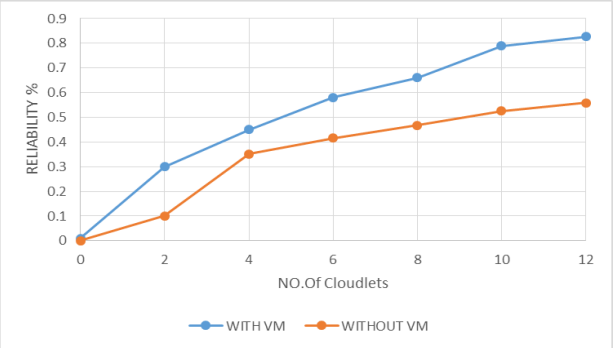

Figure-2: Comparison of VM and Host

In the Figure-2 a comparison for a virtual machine and Host is provided.

VI.CONCLUSION & FUTURE WORK

The proposed model is a good option to be used as a fault tolerance mechanism for real time computing on cloud infrastructure. It has all the advantages of forward recovery mechanism. It has a dynamic behavior of reliability configuration. The scheme is highly fault tolerant. The reason behind improve the reliability is that the scheme can take advantage of dynamic scalability of cloud infrastructure. This system takes the full advantage of using diverse software. In this experiment, we have used three virtual machines. It utilizes all of three virtual machines in parallel. This scheme has incorporated the concept of fault tolerance on the basis of VM algorithm reliability. Decision mechanism shows convergence towards the result of the algorithm which has highest reliability. Probability of failure is very less in our devised scheme. This scheme works for forward recovery until all the nodes fail to produce the result. The system change the reliability by providing the backward recovery at two levels. First backward recovery point is Time Checker. Here if all the nodes fail to produce the result, it performs backward recovery. Second backward recovery point is Decision mechanism. It performs the backward recovery if the node with best reliability could not achieve the System Reliability Level (SRL). There is another big advantage of this scheme. It does not suffer from domino effect as check pointing is made in the end when all the nodes have produced the result. The reliable VM identification technique used in this process is very efficient to improve the QOS of cloud. In future some enhancements to this model has to he done so that our system could be more fault-tolerant.

REFERENCES

[1] "Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing" Anju Bala, Inderveer Chana.

[2] X. Kong, J. Huang, C. Lin, P. D. Ungsunan, " Performance, Fault-tolerance and Scalability Analysis of Virtual Infrastructure Management System", 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, Chengdu, China, August 9-12, 2009

[3] Z. Dai, F. Viale, X. Chi, D. Caromel, Z. Lu, "A Task-Based Fault-Tolerance Mechanism to Hierarchical Master/Worker with Divisible Tasks", 2009 11th IEEE International Conference on High Performance Computing and Communication, Seoul Korea, June 25-27 2009

[4] D. Caromel, C. Delbe, A. D. Costanzo, Peer-to-Peer and fault-tolerance: Towards deployment-based technical Services, Vol 23, No. 7, August 2007, pp. 879-887

[5] X. Kong, J. Huang, C. Lin, "Comprehensive Analysis of Performance, Fault-tolerance and Scalability in Grid Resource Management System", 2009 Eighth International Conference on Grid and Cooperative Computing, Lanzhou, China, August 27-29, 2009

[6] J .Coenen, J. Hooman, "A Formal Approach to Fault Tolerance in Distributed Real-Time Systems", Department of Mathematics and Computing Science, Eindhoven University of Technology, Nether land

[7]Antonina Litvinova, Christian Engelmann and Stephen L. Scott," A Proactive Fault Tolerance Framework for High Performance Computing", 2009.

[8]Golam Moktader Nayeem, Mohammad Jahangir Alam," Analysis of Different Software Fault Tolerance Techniques", 2006.

[9]Steven Y. Ko, Imranul Hoque, Brian Cho and Indranil Gupta, "On Availability of Intermediate Data in Cloud Computations", 2010.

[10]Elvin Sindrilaru,,Alexandru Costan,, Valentin Cristea," Fault Tolerance and Recovery in Grid Workflow Management Systems", 2010 International Conference on Complex, Intelligent and Software Intensive Systems.

[11] M.Armbrust, A.Fox, R. Griffit,et al., "A view of cloud computing", Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[12] Webbing Zhao et. al. "Fault Tolerance Middleware for cloud computing." Third International Conference on Cloud Computing (2010): 67-74.

[13] Tchana Alain et. al. "Fault Tolerant Approaches in Cloud Computing Infrastructures." The Eight International Conference on Autonomic and Autonomous Systems (2012): 42-48.

[14] Slawinska, Magdalena, Jaroslaw Slawinski, and Vaidy Sunderam. "Unibus: Aspects of heterogeneity and fault tolerance in cloud computing." 2010 IEEE International Symposium on Parallel Distributed Processing Workshops and Phd Forum IPDPSW 2 (2010): 1-10.

[15] H. Chen, G. Jiang, and K. Yoshihira. "Failure detection in large-scale internet services by principal subspace mapping." IEEE Trans. on Knowledge and Data Engineering, (2007).

[16] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer. Failure diagnosis using decision trees. Autonomic Computing, International Conference on Autonomic Computing (ICAC), (2004)

[17] O. S. Unsal, I. Koren, M. Krishna, "Towards Energy-Aware Software Based Fault Tolerance in Real Time Systems"ISLPED '02, Monterey, California, USA, August 12-14, 2002,

[18] S. Malik, M. J. Rehman, "Time Stamped Fault Tolerance in Distributed Real Time Systems"; IEEE International Multitopic Conference, Karachi, Pakistan, 2005

[19] L. L. Pullum, "Software Fault Tolerance and Implementation" Artech House, Boston, London, United Kingdom, 2001

[20] K. H. Kim, "Structuring DRB Computing Stations in Highly Decentralized Systems,: Proceedings International Symposium on Autonomous Decentralized Systems, Kawasaki, 1993, pp. 305-314

[21]L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A break in the clouds: towards a cloud definition," SIGCOMM Computer Communication Review,vol. 39, pp. 50–55, December 2008.

[22]R.Buyya, S.Pandey and C.Vecchiola, "Cloudbus toolkit for market-oriented cloud computing", In Proceeding of the 1st International Conference on Cloud Computing (CloudCom2009), Beijing, China, December 2009.

[23]S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, and A. D. Keromytis, "ASSURE: Automatic Software Self-healing Using REs-cue points", Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'09), ACM Press, March 7-11, 2009, Washington, DC, USA, pp.37-48.

[24]B. Buck and J. K. Hollingsworth, "An API For Runtime Code Patching", International Journal of High Performance Computing Ap-plications, Vol.14, No.4, November 2000, pp.317-329.

[25]S. Osman, D. Subhraveti, G. Su, and J. Nieh, "The Design and Implementation of Zap: A System For Migrating Computing

Environments", Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02), USENIX Association, December 9-11, 2002, Boston, Massachusetts, USA, pp.361-376.
[26]Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou, Weizhong Qiang, Gang Hu, "SHelp: Automatic Selfhealing for Multiple Application Instances in a Virtual Machine Environment", IEEE International Conferenceon Cluster Computing, 2010.
[27] K. H. Kim, "Towards Integration of Major Design Techniques for Real-Time Fault-Tolerant Computer System", Society for Design & Process Science, USA, 2002
[28] K. H. Kim, "Distributed Execution of Recovery Blocks: An Approach to Uniform Treatment of Hardware & Software faults." Proceeding fourth International Conference on Distributed Computing Systems, 1984, pp. 526-532
[29] D. Caromel, C. Delbe, A. D. Costanzo, M. Leyton, ProActive: An Integrated Platform for Programming and Running Applications on Grids and P2P Systems, Computational Methods in Science and Technology 12(1), pp 69-77, 2006
[30] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling And Simulation Of Scalable Cloud Computing Environments And The Cloudsim Toolkit: Challenges And Opportunities," Proc. Of The 7th High Performance Computing And Simulation Conference (HPCS 09), IEEE Computer Society, June 2009.