International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
2nd National Conference on Innovative Paradigms in Engineering & Technology (NCIPET 2013) – www.ijais.org

Fault Tolerance Model in Cloud Computing

Anjali.D.Meshram
Dept.of Computer Sci. & Engg.
P.I.E.T
Nagpur

A.S.Sambare
Dept.of Computer Technology.
P.I.E.T
Nagpur

S.D.Zade
Dept.of Computer Sci. & Engg.
P.I.E.T
Nagpur

ABSTRACT
Cloud computing has emerged as a platform that grants users with direct yet shared access to remote computing resources and services. Cloud must provide services to many users at the same time; the scheduling strategy should be developed for multiple tasks. In cloud computing processing is done on remote computer hence there are more chances of errors, due to the undetermined latency and loose control over computing node. Hence remote computers should be highly reliable. This is reason why a cloud computing infrastructure should be fault tolerant as well scheduling properly to performing tasks. This paper mainly deals with a fault tolerance model for cloud computing & Paper describes model for Fault Tolerance in Cloud computing (FTMC) FTMC model tolerates the faults on the basis of reliability of each computing node. A Computing node is selected for computation on the basis of its reliability and can be removed, if does not perform well for applications.

General Terms
Cloud Computing, Fault Tolerance

Keywords
Cloud Computing, Fault Tolerance, and Reliability

1. INTRODUCTION
Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). . Cloud computing emerges as a new computing paradigm which aims to provide reliable, customized and QoS (Quality of Service) guaranteed computing dynamic environments for end-users. Overall, cloud computing brings new aspects in computing resource management: infinite computing resources available on demand for the perspective of the end users; zero up-front commitment from the cloud users; and short-term usage of any high-end computing resources [11], [12].The basic principle of cloud computing is that user data is not stored locally but is stored in the data center of internet. The companies which provide cloud computing service could manage and maintain the operation of these data centers. The users can access the stored data at any time by using Application Programming Interface (API) provided by cloud providers through any terminal equipment connected to the internet. Not only are storage services provided but also hardware and software services are available to the general public and business markets. The services provided by service providers can be everything, from the infrastructure, platform or software resources. Each such service is respectively called Infrastructure as a Service (IaaS), Platform as a Service (PaaS) or Software as a Service (SaaS).To give services to end users a cloud computing environment should be reliable & should mange to give output in minimum amount of time. Hence fault tolerance & task scheduling are the two important parameters for a system to be reliable. Cloud Computing environment should able to work on two basic characteristics that is Timeliness & Fault tolerance. By timeliness, we mean that each task in real time system has a time limit in which it has to finish its execution. And by fault tolerance means that it should continue to operate under fault presence

2. RELATED WORK
Cloud computing is gaining an increasing popularity over traditional information processing systems for storing and processing huge data. This computing model is built on modern data centers made up of thousands of interconnected servers with capability of hosting a large number of applications.[8] Most often, these data centers are virtualized, and computing resources are provisioned to the user as an on demand services over the Internet in the form of configurable Virtual Machines (VMs) [9]. Cloud infrastructure should be fault tolerance. Lot of work has been done up till date to make cloud infrastructure fault tolerant. In cloud computing latency of node or virtual machine is unknown, as it always changes over a time.[2].User of a cloud do not know at exactly which node his request is being processed. but one advantage of a cloud computing is that it has capability to scale up dynamically. So that any node which is not working properly can be removed &also new node can be added to get maximum output from cloud. X. Kong et. al. [1, 2] presented a model for virtual infrastructure performance and fault tolerance. But it is not well suited for the fault tolerance of real time cloud applications. A model for non-cloud environment is proposed by J. Coenen and J.Hooman [3] which describes a model for fault tolerance in distributed real time system. Ravi Jhawar, Vincenzo Piuri, Marco Santambrogio in their paper [4] "A Comprehensive Conceptual System-Level Approach to Fault Tolerance in Cloud Computing", describes a framework for user in which user can specify and apply the desired level of fault tolerance without requiring any knowledge about its implementation with the help of dedicated user service layer. Another model is "time stamped fault tolerance of distributed RTS" [5], which is proposed by S. Malik and M. J. Rehman. This model proposed time stamping with the outputs. All these models mainly deal with the fault tolerance without taking reliability of computer nodes into consideration. Fault Tolerant system means system should work under the presence of fault. [4]. Techniques to build efficient and fault tolerant applications for Amazon's EC2 are provided in [6]. Another approach using fault tolerance m middleware which follows a leader/follower replication approach to tolerate crash faults has been proposed in [7]. However, all these techniques either

tolerate only a specific kind of fault or provide a single method to resilience. The reliability of cloud system is a major concern among users. In "Approach for constructing a modular Fault tolerant protocol "paper by M.Hiltunen & R.D. Schlichting proposed a modular protocols & combining them to a system using hierarchical techniques.[10].

3. PROPOSED SYSTEM

The proposed system deals with the fault tolerance mechanism In this proposed system , a model name fault tolerance model for cloud (FTMC) model is based upon reliability assessment of computing nodes known as a virtual machines(VM) in cloud environment and fault tolerance of real time applications running on those VMs. A virtual machine is selected for computation on the basis of its reliability and can be removed, if does not perform well for real time applications. In this scheme, 'N' virtual machine, which run the 'N' variant algorithms Algorithm 'X1' runs on 'Virtual machine-1', 'X2' a run on 'Virtual machine-2', up till 'Xm', which runs on 'Virtual machine m'. Then there is Accepter which is responsible for the verification of output result of each node. The outputs are then passed to Timer which checks the timing of each result. On the basis of the timing the Reliability Assessor calculates and reassigns the reliability of each module. Then all the results are forwarded to Decision Maker which selects the output on the basis of best reliability. The output of a node with highest reliability is selected

3.1 Working of Model

In the fault tolerance mechanism, has M virtual machines. Each node is taking input data from the input buffer. This input is concurrently passed to all the virtual machines. Each node takes the input, executes the application algorithm and produces result. These results are passed to ACCEPTER module. ACCEPTER module then passes these results to the assembly node for result decision and reliability assessment. The different modules in the system have the given responsibility. As we stated earlier that all the VMs are running different real time algorithms. These algorithms are different from each other by their implementation language, logics, software, operating systems etc.

Accepter module is provided for each VMs.It performs testing on VMs for each iteration. Testing is carried out on algorithms running on each virtual machine. If a given algorithms gives desired result then it further passes to assembly nodes. If result is incorrect then it does not passes it to assembly nodes. Assembly nodes consist of Timer module, Reliability Assessor (RA) module & Decision Maker (DM) module.

In Timer module timer is set for each VM.It monitors the timing results produced by each VM. It receives the output of ACCEPTER module & passes only those correct results from a node which produces before time.

Reliability Assessor (RA) module assesses the reliability for each virtual machine. The reliability of the virtual machine changes after every computing cycle. At the start, the reliability of each virtual machine is 100%. If a processing VM manages to produce a correct result within the time limit, its reliability increases. And if the processing VM fails to produce the correct result or result within time, its reliability decreases. Every virtual machine has its predefined limit value for maximum & minimum reliability. If reliability of any processing VM falls below minimum reliability value then RA will restrict that node from working & removes it by sending a message to resource manager to remove it &add new node in place of that particular node. If reliability of all the nodes fall below minimum reliability level, the system either perform the backward recovery or stop the system to work further. The output of RA will then further passes to DM module.

Decision Maker (DM) module takes the input from RA module. It selects the output of the node which has highest reliability among all the competing nodes. Competing nodes are those nodes which have produced the results within There is a SRL (system reliability level). It is the minimum reliability level to be achieved to pass the result. DM compares the best reliability with the system reliability level. Best reliability level should be greater than or equal to time. system reliability level. If the node with best reliability does not achieve the SRL the DM raises the failure signal for the computing cycle. In this case, backward recovery is performed. Backward is performed by the help of check point established in recovery cache. DM also request to some responsible module (resource manager or scheduler) to remove one node with minimum reliability and add a new node.

Check Point (CP) is the points from where backward error recovery is done in case of complete failure of a system. 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.

3.2 Fault Tolerance Mechanism

Reliability assessment algorithm is applied on each node (virtual machine) one by one. Initially reliability of a node is set to 1. There is an adaptability factor ***n***, which controls the of reliability assessment. The value of *n* is always greater than 0. The algorithm takes input of three factors *RF*, minReliability and maxReliability from configuration file. *RF* is a reliability factor which increases or decreases the reliability of the node. minReliability is the minimum reliability level. If a node reaches to this level, it is stopped to perform further operations. maxReliability is the maximum reliability level. Node reliability cannot be more than this level. 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 real time applications. User has to decide how much be the value for each variable.

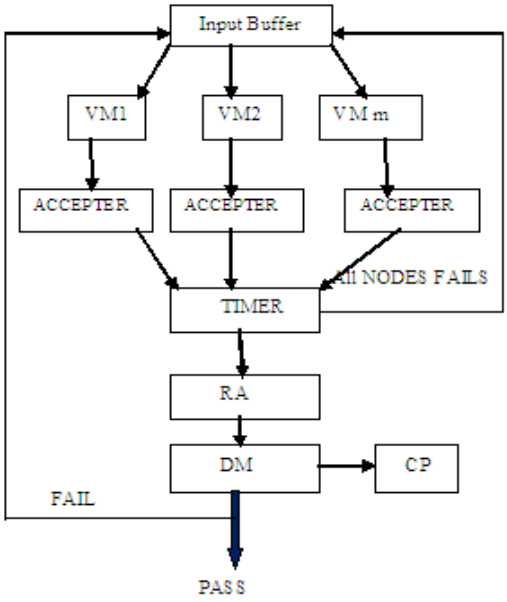

Fig 1: Proposed Fault Tolerance system

Reliability Assessment Algorithm:

Begin

Initially reliability: =1, n: =1

Input from configuration RF, maxReliability,

MinReliability

Input node status

if node Status =Pass **then**

Reliability: = reliability + (reliability * RF)

if n > 1 **then**

n := n-1;

else if node Status = Fail **then.**

reliability := reliability – (reliability * RF * n)

n := n+1;

if reliability >= maxReliability **then**

Reliability: = maxReliability

if reliability < minReliability **then**

Node Status: =dead

call_proc: remove_this_node

call_proc: add_new_node

End.

Decision Mechanism Algorithm

Begin

Initially reliability:=1, n :=1

Input *from RA* node Reliability, num C and Nodes

Input *from configuration* SRL

best Reliability: = find reliability of node with highest

reliability

if best Reliability >= SRL

status := success

else

perform_backward_recovery

call_proc: remove_node_minReliability

call_proc: add_new_node

End.

3.3 Different Scenarios

3.3.1) Complete Failure Free Scenario

All the algorithms on each virtual machine produce the result. Accepter module passes the results. All the results are produced before time overrun. So Timer also clears the results. RA computes and assigns the new reliability weights to each virtual machine. Decision mechanism selects an output from the VM with maximum reliability. No failure or exception occurs in any VM in this case.

3.3.2) Partial Failure Scenario – All Accepter pass, timer passes some Nodes

All the virtual machines produce the correct result. Some results are produced within time and some after time overrun. All Accepter pass the results and forward them to the timer checker. Time receives result of some virtual machines before time-overrun. It passes them to RA, which assesses their reliability. RA forwards the produced result to the decision maker.. Decision maker selects an output from the VM with maximum reliability. In this scenario, system will continue to operate with forward recovery.

3.3.3) Partial Failure Scenario – Some Accepter pass, timer also Pass

Some of the virtual machines produce the correct result. These results are produced within time limit. Accepter passes only the correct results to the Timer. For failed virtual machines, it generates an error signal to Timer. Timer receives result of passed virtual machines before time overrun. It passes them to RA, which assesses their reliability. RA forwards the

produced result to the decision maker. Decision mechanism selects an output from the VM with maximum reliability. In this scenario, system will continue to operate with forward recovery.

3.3.4) Failure Scenario – Accepter s fail, Timer fail

In this scenario, either all the Accepter rejects the result of the algorithms or some Accepter passes but Timer fails to find a single output within time limit. In this case, the cycle fails and Timer informs the Assembly nodes to perform backward recovery. Now backward recovery will be done with the help of checkpoints

3.3.5) Failure Scenario – Accepters pass, Timers pass, and DM fails

In this scenario, all or some of the Accepter passes the results. Timer also finds the output within time limit. Reliability assessor computes and assigns the reliability to the virtual machines. But the VM with highest reliability could not achieve the system reliability level (SRL). In this case, DM raises the failure signal for the whole computing cycle and backward recovery is performed with the help of checkpoint stored in recovery cache.

4. DISCUSSION & CONCLUSION

The proposed scheme is a good option to be used as a fault tolerance mechanism for computing on cloud infrastructure. This scheme has incorporated the concept of fault tolerance on the basis of VM algorithm reliability. Probability of failure is very less in proposed scheme. This scheme works for forward recovery until all the nodes fail to produce the result. The system assures the reliability by providing the backward recovery at two levels. First backward recovery point is TC. Here if all the nodes fail to produce the result, it performs backward recovery. Second backward recovery point is DM. It performs the backward recovery if the node with best reliability could not achieve the 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 results

5. FUTURE SCOPE

We can further make cloud environment more faults tolerant by including more parameters in decision maker module.

6. REFERENCES

[1] 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.

[2] 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

[3] 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

[4]Ravi Jhawar, Vincenzo Piuri, Marco Santambrogio," A Comprehensive Conceptual System-Level Approach to Fault Tolerance in Cloud Computing", © 2012 IEEE, DOI 10.1109/SysCon.2012.6189503

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

[6] J. Barr, A. Narin, and J. Varia, "Building Fault-Tolerant Applications on AWS," October 2011. <Online>Available:http://media.amazonwebservices.com /AWS-Building-Fault-tolerant-application.pdf

[7] W. Zhao, P. M. Melliar-Smith, and L. E. Moser, "Fault Tolerance Middleware for Cloud Computing," in Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, ser. CLOUD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 67–74.

[8] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in Proceedings of the 1st ACM symposium on Cloud computing, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 193–204.

[9]Amazon elastic computes cloud. http://aws.amazon.com/ec2/.

[10] M.Hiltunen & R.D. Schlichting "Approach for Constructing a Modular Fault -Tolerant Protocols " in In Proceedings of the 12th Symposium on Reliable Distributed systems, IEEE,1993,pp,105-114.

[11] M. Armbrust, A. Fox, and et al., "Above the clouds: A Berkeley view of cloud computing," UC Berkeley, Tech. Rep. UCB/EECS-2009-28, February 2009.

[12] K. Birman, G. Chockler, and R. van Renesse, "Toward a cloud computing research agenda," *SIGACT News*, vol. 40, no. 2, pp. 68–80, 2009.