ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Fault Tolerance in Cloud Computing: A Review

[1]Gagan Amrit Kaur, [2]Sonia Sharma
[1,2]Dept. of Computer Engineering & Technology, G.N.D.U, Amritsar, Punjab, India

Abstract
Cloud computing may be defined as management and provision of resources, software, application and information as services over the cloud which are dynamically scalable. The dynamic environments of cloud are more or less prone to failure. So there is an increased requirement for fault tolerance to achieve reliability for the real time computing on cloud infrastructure. Fault tolerance techniques are used to predict these failures and take an appropriate action before failures actually occur. Various fault detection methods have been discussed. In this paper, a fault tolerance in real time cloud computing Environment is discussed in which the system tolerates the faults and makes the decision on the basis of reliability of the processing nodes, i.e. virtual machines.

Keywords
Fault Tolerance, Reliability, Cloud Computing, Virtual Machines, Real-Time

I. Introduction
Cloud computing is a style of computing where service is provided across the Internet using different models and layers of abstraction [1]. It refers to the applications delivered as services [2] to the mass, ranging from the end-users hosting their personal documents on the Internet to enterprises outsourcing their entire IT infrastructure to external data centers.
A simple example of cloud computing service is Yahoo email or Gmail etc.

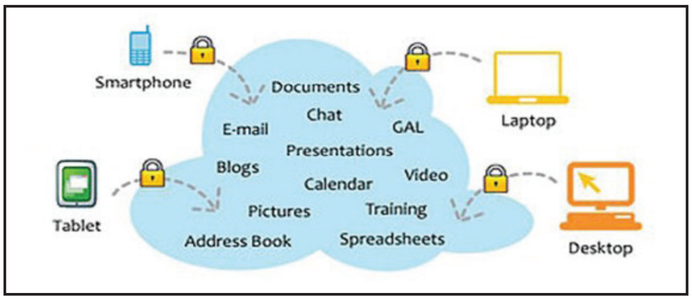


Fig. 1: Cloud Computing

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 [3-4]. The basic principle of cloud computing is that user data is not stored locally but is stored in the data centre of internet.

A. Cloud Components
Cloud computing is made up of several elements. Each element has a purpose which plays specific roles which can be classified as clients, Distributed servers, data centers.

1. Clients
These are typically the computers which are used by the end users i.e. the devices which can be used by the end user to manage the information on cloud (laptops, mobile phones, PADs etc.)

2. Data Center
These are collection of servers where the service is hosted. In order to create number of virtual server on one physical server in data center, virtualization is used.

3. Distributed Servers
These are servers which are located in different geographical place. It provides better accessibility, security to the user.

B. Essential Characteristics
There are ten main characteristics of cloud computing, summed up below [5] :

1. Scalability and on-demand services
Users are given on-demand resources and services over cloud. Moreover the resources provided are scalable over several data centers.

2. User-centric Interface
Cloud interfaces are not dependent on location of user. They can be accessed by well-established interfaces such as web services and internet browsers.

3. Guaranteed Quality of Service (QOS)
Cloud computing assures Quality of service for users by guaranteed performance, bandwidth and memory capacity.

4. Autonomous System
Users can reconfigure and combine software and information according to their requirements.

5. Cost
No capital expenditure or any up-form investment is required in cloud. Payment for services is made on the basis of need.

6. Broad Network Access
Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

7. Resource Pooling
The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. Examples of resources include storage, processing, memory, and network bandwidth.

8. Virtualization
Utilization of resources is increased by sharing the server and storage devices.

9. Multi-tenancy
Sharing of resource and cost among large number of users increase efficiency and allows for centralization and peek lock capacity.

10. Loose Coupling
The resources are loosely coupled as one resource functionality hardly affects the functioning of another resource.

C. Service Models
Services offered by the cloud providers can be grouped into three categories [6]:

1. Software as a Service (SaaS)
In this model, a complete application is provided on demand to the user. Multiple end users are serviced while at the back end a single instance of service is executed. Customers need not to go for any upfront investments, since just a single application is to be facilitated & kept up. Google, Salesforce, Microsoft etc are the providers of Saas.

2. Platform as a Service (Paas)
In this model, software or development environment is offered as a service. The customer is given with the option to construct his own particular applications, which run on the suppliers' base. A predefined combination of OS and application servers is provided to the user. Google's App Engine, Force.com are providing a platform to users.

3. Infrastructure as a Service (Iaas)
Standardized services that are provided are Fundamental storage and computing capabilities. Various resources are made available and shared among users in order to manage workload. The customer has to deploy his own software on the infrastructure. Amazon, GoGrid, are examples of Iaas.

D. Deployment Models
On the bases of access to clouds, they can be classified into following types [6]:

1. Public Cloud
Users connected to internet and having access to the cloud space can use public cloud. It refers to availability of computing resources to anyone on "Pay As You Go Basis". Public clouds are owned and operated by third parties; they deliver superior economies of scale to customers. All customers share the same infrastructure pool with limited configuration, security protections, and availability variances.

2. Private Cloud
A private cloud in an organization is specific and limited access to a particular group. It can be referred as computing services delivered exclusively for the use of a particular organization or a group. It utilizes the same architecture for scalability and availability as the public cloud but it is limited to a single organization. Two major concerns on data security and control are addressed which are not there in public cloud.

3. Hybrid Cloud
A combination of public and private cloud is named as hybrid cloud. With a Hybrid Cloud, service providers can expand the adaptability of computing by utilizing other Cloud Providers in full or partial manner. The Hybrid cloud environment is capable for providing on-demand, externally provisioned scale with the capacity to enlarge a private cloud to deal with any sudden surges in workload.

4. Community Cloud
The organizations with common prerequisites share the cloud functionality making it a hybrid cloud. It reduces the capital consumption by imparting the cost among the associations. The operation may be in-house or with an outsider on the premises.

II. Fault Taxonomy
Fault Tolerance alludes to a methodology to system design that permits a system to keep performing actually when one of its parts falls flat or it can be defined as capacity of a system to react nimbly to an unexpected equipment or programming break down. If not fully operational, fault tolerance solutions may allow a system to continue operating at reduced capacity rather than shutting down completely following a failure [7].
There are various faults which can occur in cloud computing .Based on fault tolerance policies various fault tolerance techniques can be used that can either be task level or workflow level .

A. Reactive Fault Tolerance
Reactive 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 [8].

1. Check pointing/ Restart - When a task fails, it is allowed to be restarted from the recently checked pointed state rather than from the beginning. It is an efficient task level fault tolerance technique for long running applications.
2. Replication-Various task replicas are run on different resources, for the execution to succeed till the entire replicated task is not crashed. It can be implemented using tools like HAProxy, Hadoop and AmazonEc2 etc.
3. Job Migration-During failure of any task, it can be migrated to another machine. This technique can be implemented by using HAProxy.
4. SGuard- It is less disruptive to normal stream processing and makes more resources available. SGuard is based on rollback recovery and can be implemented in HADOOP, Amazon EC2.
5. Retry-It is the simplest task level technique that retries the failed task on the same cloud resource.
6. Task Resubmission-It is the most widely used fault tolerance technique in current scientific workflow systems. Whenever a failed task is detected, it is resubmitted either to the same or to a different resource at runtime.
7. User defined exception handling-In this user specifies the particular treatment of a task failure for workflows.
8. Rescue workflow-This technique allows the workflow to continue even if the task fails until it becomes impossible to move forward without catering the failed task.

B. Proactive Fault Tolerance
The principle of proactive fault tolerance policies is to avoid recovery from faults, errors and failures by predicting them and proactively replace the suspected components other working components. Some of the techniques which are based on these policies are Preemptive migration, Software Rejuvenation etc [8].

- **Software Rejuvenation-**It is a technique that designs the system for periodic reboots. It restarts the system with clean state.
- **Proactive Fault Tolerance using Self-Healing-** When multiple instances of an application are running on multiple virtual machines, it automatically handles failure of application instances.
- **Proactive Fault Tolerance using Preemptive Migration-** Preemptive Migration relies on a feedback-loop control mechanism where application is constantly monitored and analyzed.

C. Failure Detector

A failure detector is an application or a system that is used to detect node failures or crashes. Failure detector can be classified as reliable or unreliable on the basis of result it produces. If the output of failure detector is always accurate it is called as reliable failure detector. An unreliable failure detector is one that provides information that is not necessarily accurate and it may take very long time for detection of faulty process and produce false results by suspecting the processes that have not crashed. Most of the failure detectors fall in this category [9].

A. Correctness Properties of Failure Detectors

1. **Completeness:** When a process fails that process is eventually detected by at least one other non-faulty process. Completeness describes the capability of failure detector of suspecting every failed process permanently.
2. **Accuracy:** There are no mistaken failure detections i.e. when a process is detected as failed, it has actually failed. Less number of false positives result in high accuracy. It is impossible to build a failure detector over a realistic network that is 100% accurate and complete. Real life failure detectors guarantee 100% completeness but the accuracy is either partial or probabilistic. There is a trade-off between completeness and accuracy
3. **Speed:** Time for the detection of failure should be as less as possible. In other words, time between occurrence of a failure and its prediction must be small.
4. **Scale:** There should be low and equally distributed load on each process in a group and also low overall network load.

A failure detector should guarantee all of these properties in spite of the fact that there can be arbitrary simultaneous multiple process failures.

- **Detection time (TD):** Time that elapses from crashing of a process p to the time when another process q starts suspecting process p permanently.
- **Mistake recurrence time (TMR):** Time between two successive mistakes.
- **Mistake Duration (TM):** Time taken by a failure detector to correct the mistake.

Failure detectors that adapt themselves to the changing network conditions and application requirements are named as adaptive failure Detectors. Most adaptive failures are based on heartbeat protocol where previous information is used for the prediction of arrival time of next heartbeat.

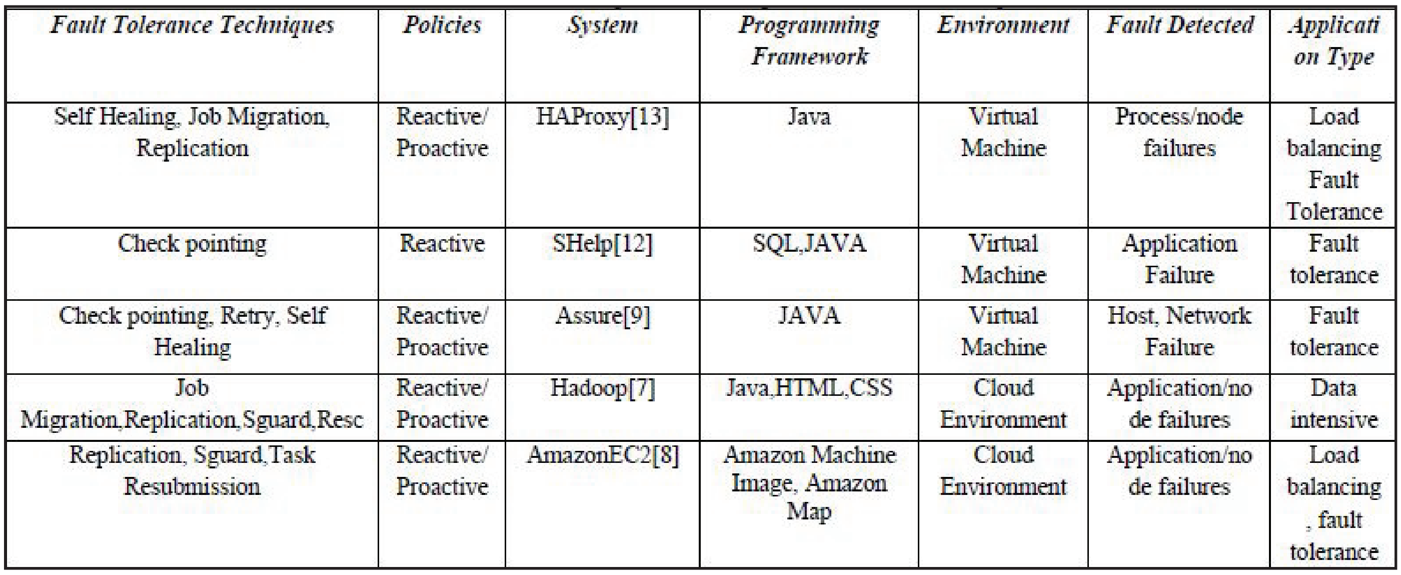
2.4 Heartbeat Strategy for Failure Detection

Heartbeat is a widely implemented strategy for failure detectors. After a fixed interval of time every process p send "I am alive" message to a process q. q waits for the message from p till the expiration of timeout from p and if the message is not received it adds p to list of suspected processes. If q later receives "I am alive" message from p, it will remove the process p from list of suspected processes.

E. Chart of Tools Used For Implementing Fault Tolerance

Fault tolerance challenges and techniques have been implemented using various tools. Table 1 [8] compares these tools based on their programming framework, environment and application type along with different fault tolerance techniques. HAProxy is used for server failover in the cloud. SHelp [12] is a lightweight runtime system that can survive software failures in the framework of virtual machines. It may also work in cloud environment for implementing check pointing. ASSURE introduces rescue points for handling programmer anticipated failures. Hadoop is used for data intensive applications but can also be used to implement fault tolerance techniques in cloud environment. Amazon Elastic Compute Cloud (EC2) provides a virtual computing environment to run Linux-based applications for fault tolerance.

Table 1: Tools Used To Implement Existing Fault Tolerance Techniques



Fault Tolerance Techniques	*Policies*	*System*	*Programming Framework*	*Environment*	*Fault Detected*	*Applicati on Type*
Self Healing, Job Migration, Replication	Reactive/ Proactive	HAProxy[13]	Java	Virtual Machine	Process/node failures	Load balancing Fault Tolerance
Check pointing	Reactive	SHelp[12]	SQL,JAVA	Virtual Machine	Application Failure	Fault tolerance
Check pointing, Retry, Self Healing	Reactive/ Proactive	Assure[9]	JAVA	Virtual Machine	Host, Network Failure	Fault tolerance
Job Migration,Replication,Sguard,Resc	Reactive/ Proactive	Hadoop[7]	Java,HTML,CSS	Cloud Environment	Application/no de failures	Data intensive
Replication, Sguard,Task Resubmission	Reactive/ Proactive	AmazonEC2[8]	Amazon Machine Image, Amazon Map	Cloud Environment	Application/no de failures	Load balancing , fault tolerance

III. Related Work

Lot of work has been done up till date to make cloud infrastructure fault tolerant. A scheme [11],[13] is devised here which is for the fault tolerance of real time applications running on cloud infrastructure. The model name is Fault Tolerance in Cloud computing (FTC). This scheme tolerates the faults on the basis of reliability of each computing node, i.e. virtual machine. 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. The model is shown in figure 2. In this model, we have two main types of node. One type is a set of virtual machines, running on cloud infrastructure, and the other is the adjudication node. Virtual machine contains the real time application algorithm and an acceptance test for its logical validity. On the adjudicator, we have the time checker, reliability assessor and decision mechanism modules. This scheme provides forward recovery as well as optional backward recovery. In this scheme, we have 'N' virtual machine, which run the 'N' variant algorithms. Algorithm 'X1' runs on 'Virtual machine-1', 'X2' runs on 'Virtual machine-2', up till 'Xm', which runs on 'Virtual machine-m'. Then we have AT acceptance test module which is responsible for the verification of output result of each node. The outputs are then passed to TC time checker module which checks the timing of each result.

On the basis of the timing the RA reliability assessor module calculates and reassigns the reliability of each module. Then all the results are forwarded to DM decision mechanism module which selects the output on the basis of best reliability. The output of a node with highest reliability is selected as the system cycle output.

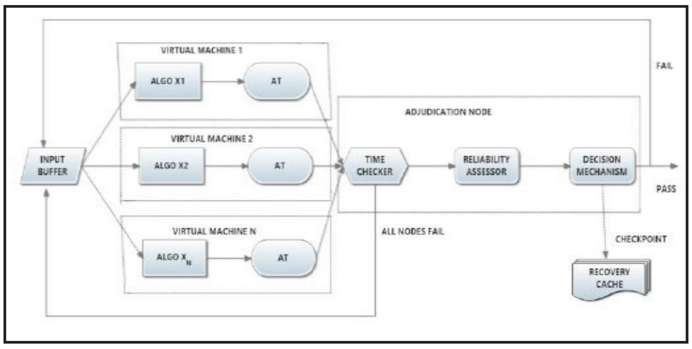


Fig. 2: Fault Tolerance in Cloud Computing

A. Fault Tolerance Mechanism

Here we apply reliability assessment algorithm for 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 adaptability 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. It decreases the reliability of the node more quickly as compare to the increase in reliability. It is due to its multiplication with the adaptability factor n. 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. 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 past. 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 real time applications. User has to decide how much be the value for each variable. Calculation of these variables is not within the scope of this research.

B. Reliability Assessment Algorithm

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

C. Reliability Assessment Impact Analysis

In the Table 1, a comparison is provided between pass and fail scenario.

Table 1: Comparison of Pass & Fail

Cycle	Status	Reliability	Status	Reliability
Start	-	1.0000	-	1.0000
1	Pass	1.0300	Fail	0.9700
2	Pass	1.0609	Fail	0.9118
3	Pass	1.0927	Fail	0.8297
4	Pass	1.1255	Fail	0.7302
5	Pass	1.1593	Fail	0.6206
6	Pass	1.1941	Fail	0.5089
7	Pass	1.2299	Fail	0.4021
8	Pass	1.2668	Fail	0.3056
9	Pass	1.3048	Fail	0.2231
10	Pass	1.3439	Fail	0.1561

This comparison is done for 10 computing cycle. In these cycles a node continuously passed and another node continuously failed. The increase in reliability after 10 cycles for VM-1 is 0.3439, whereas decrease in reliability for VM-2 is 0.8439. Here we can see that decrease due to failure is more than increase. And this decrease continues to decrease faster if the node continues to fail.

References

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

[2] 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.

[3] 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.

[4] K.Birman, G.Chockler, R.van Renesse,"Toward a cloud computing research agenda," SIGACT News, Vol. 40, No. 2, pp. 68–80, 2009.

[5] Gong, C., Liu, J., Zhang, Q., Chen, H., Gong, Z., The characteristics of cloud computing", In Parallel Processing Workshops (ICPPW), 2010 39th International Conference on (pp. 275-279). IEEE, 2010.

[6] Furht, B.,"Cloud computing fundamentals", In Handbook of cloud computing (pp. 3-19). Springer US, 2010.

[7] Kaushal, V., Bala, A.,"Autonomic fault tolerance using haproxy in cloud environment", Int. J. of Advanced Engineering Sciences and Technologies, 7(2), pp. 54-59, 2011.

[8] IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[9] International Journal of Computer Applications (0975 – 8887) Vol. 116 – No. 18, April 2015

[10] International Journal of Computer Science and Communication Engineering Vol. 5, Issue 1(February 2016 issue)

[11] S. Sudha Lakshmi,"Fault Tolerance in Cloud Computing", International Journal of Engineering Sciences Research-IJESR, ACICE-2013.

[12] 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 Conference on Cluster Computing, 2010.

[13] 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).