A Comprehensive Survey of Fault Tolerance Techniques in Cloud Computing

Himanshu Agarwal, Anju Sharma
Computer Science and Engineering Department
Thapar University
Patiala, India

***Abstract*— Fault tolerance in cloud computing platform is a crucial issue as it guarantees the availability, performance and reliability of the applications. In order to achieve the availability, reliability, performance, robustness and dependability in cloud computing, failure should be accessed and handled effectively. This paper discusses the better understanding of different fault tolerance techniques which are used according to their policies and tools. This paper also describes the comprehensive taxonomy of faults, errors and failures. The usage of taxonomy and survey results are not only used to identify the similarities but also to identify the areas requiring for future research.**

Keywords*— *fault tolerance; cloud computing; availability; reliability.

I. INTRODUCTION

Cloud Computing is a concept which refers to services and applications which are executed on a distributed network with the help of resources which are virtualized. Cloud computing provides an abstract view of physical systems on which the application and software are executed. It is considered that resources provided in cloud are virtual and limitless and customer can pay for only those resources which are required by them.

Quality of Service (QoS) in cloud computing plays an important role in which performance and reliability of service are two important aspects. Performance in cloud computing means if a user request for a particular service how fast that service can be provided by service provider while reliability in cloud computing means if the service requested by a user will be provided successfully or not. Both the performance and reliability plays an important role in cloud computing because if reliability of service is low, it will cause frequent failures of cloud services which in turn will reduce the number of dissatisfied customers which will cause loss to service provider, if service reliability was high but performance was low then users who have requested for services will have to wait for longer period of time which will again cause dissatisfied customer. Therefore, it is directly related with fault tolerance [1].

To improve reliability in cloud computing, fault tolerance [2-4] is used. Fault tolerance is property of a system which allows it to provide required service with possibly degraded performance if there are one or more faults, or component failures [5]. Different types of faults may occur in a cloud infrastructure and there are various fault tolerance techniques that can be used based on these fault tolerance policies.

II. RELATED WORK

Fault tolerance computing is a field that has been rapidly developing since 1970. Various journals and conferences started publishing papers based on fault tolerance e.g. IEEE Transactions on Computers, IEEE Computers, etc. Tandem-16 was the first computer for fault tolerance in online transaction processing.

In 1999, Felix C. Gartner [6] discussed fundamental methodologies and their relation on distributed computing with the help of a formal approach which helped in better understanding of the subject and build a more reliable and dependable system.

In 2004, Lee Pike et al., [7] provided four different abstractions for distributed fault tolerant systems. These four methods abstracts a wide variety of distributed fault tolerant system. These abstraction are related to faults, communication, messages and fault masking.

In 2011, Arvind Kumar et al., [8] have checked various fault tolerance techniques which can be used in real time distributed system. This paper checks how different fault tolerance techniques are applied to tolerate different faults present in real time distributed system.

In 2011, Zhang et al., [9] proposed a Byzantine fault tolerance (BFT) system for cloud which provide high reliability in cloud which also ensures high performance of these systems.

In 2013, P. K. Patra et al., [11] provided taxonomy of different fault tolerance techniques based on various metrics i.e. reliability, performance, etc.

978-1-4673-7309-8/15/$31.00 ©2015 IEEE

In 2013, N. Chandrakala and P. Sivaprakasam [12] provided a load balancing algorithm for cloud computing which checks if the load on any virtual machine in cloud is more than 75%. It transfer it to a new virtual machine whose load is less than 75% and does not exceed more than 75% after the load is transferred.

In 2013, Anjali D. Meshramet et al., [13] provided a Fault Tolerance in Cloud Computing (FTMC) model which works on the principle of reliability of each computing node. A node is only selected if its reliability is high otherwise it is removed.

In 2013, Ravi Jhawar et al., [14] proposed a comprehensive high-level approach which assess fault tolerance mechanism and use virtualization technology to improve availability and reliability of applications which are placed in virtual machines in cloud.

III. Taxonomy of Fault, Error and Failure in Cloud

A. Fault Tolerance

It is the property of a system which allows it to provide the required service (with degraded performance) if there are one or more than one faults, or component failures. Fig. 1 shows path to generation of failure.

Fig 1. Path to Generation of Failure

B. Fault

Inability of a system to do its required task caused by an anomalous state or bug in one or more than one parts of a system. Faults are the hypothesized or adjudged cause of an error i.e. the main cause which causes an error [15]. Various faults are classified as shown in Fig. 2.

C. Error

A quantity of predicted difference between the calculated or observed value of a quantity and its true value. Activation of fault may put the system component in incorrect condition. An error can be a cause of a system to perform erroneously which may result into failure of the system. Various errors are classified as shown in Fig. 3.

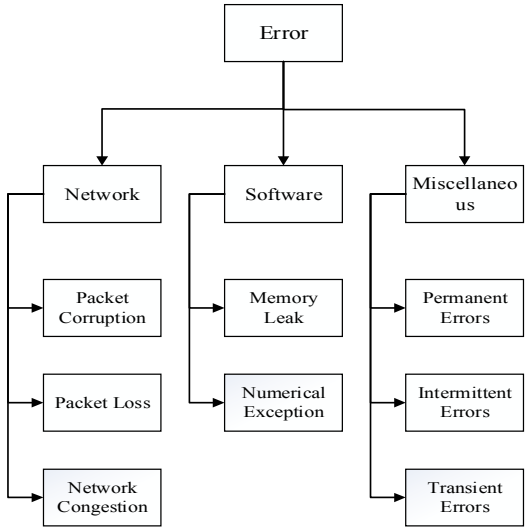

Fig. 3. Taxonomy of Errors

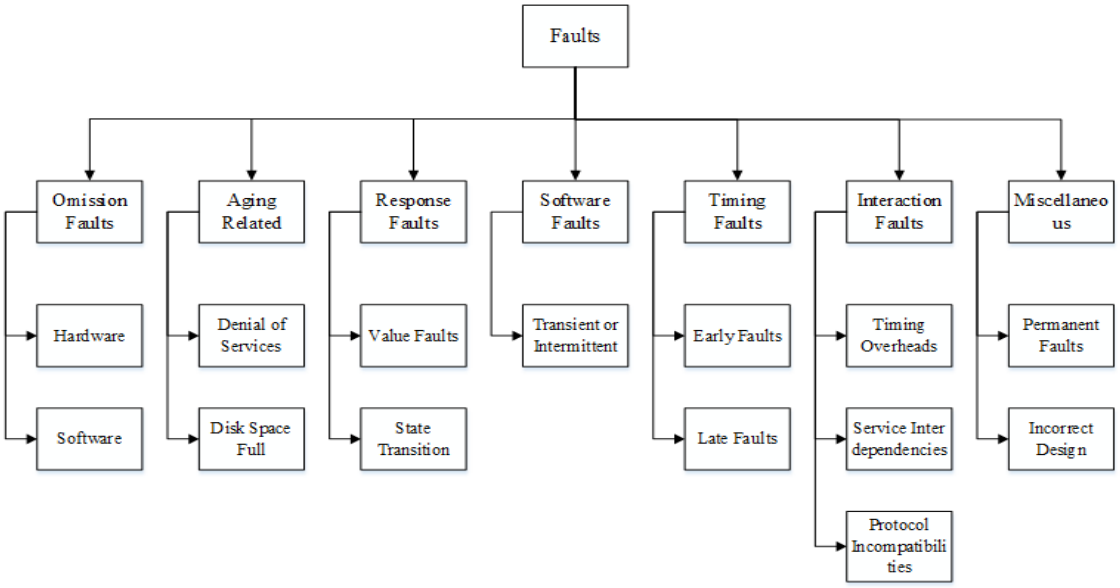

Fig. 2. Taxonomy of Faults

D. Failure

The state in which the intended function does not perform the desirable or intended objective. Failure refers to misconduct of a system which can be observed by a user, which can either be human or another computer system. Various things can go wrong inside a system, but if does not result in incorrect output there is no failure [15]. Various failures are classified as shown in Fig. 4.

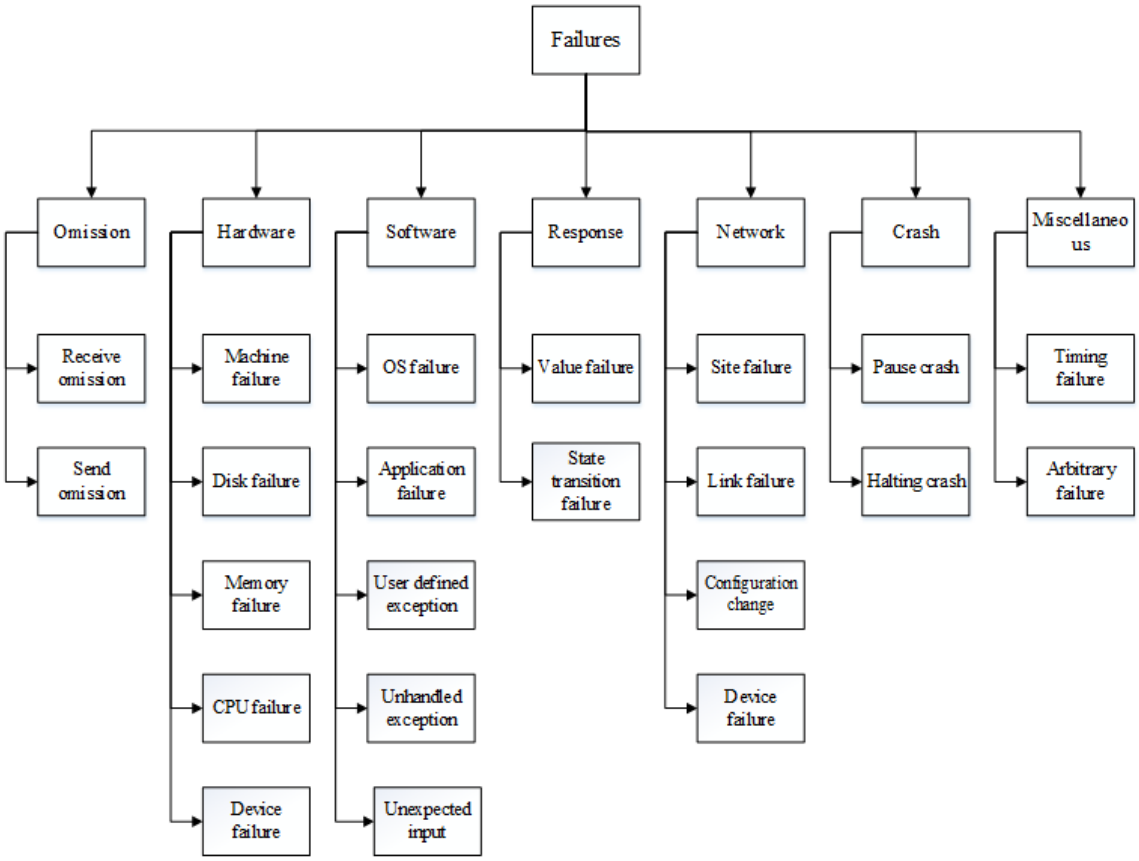

Fig. 4. Taxonomy of Failures

IV. Existing Fault Tolerance Techniques

Based on fault tolerance policies and techniques we can classify them into two types reactive and proactive. Fault tolerance techniques are further classified into various types as shown in Fig. 5.

A. Reactive Fault Tolerance

Reactive fault tolerance policies are used when a failure has occurred in the system. It helps in recovery of system state from an unstable state to stable state so that the system can again start working to provide desired results. Various reactive fault tolerance techniques are discussed below.

1) Restart: It operates on program, or application level. If a task does not complete within a given amount of time it is suspected to have failed and is, consequently, aborted and restarted [16]. The timeout after which we have to abort and restart the task must be carefully chosen because if it is too short then the task might be aborted just before completion while if it is chosen too long one must wait unnecessarily.

2) Checkpointing: It has a preventive component i.e. saving a checkpoint as well as a reactive component i.e. rollback recovery [16]. Checkpointing systems save the system state in regular or irregular time intervals. Upon failure the system recovers by rolling back to most recent checkpoint. The work performed since the most recent checkpoint is lost with a failure. Checkpointing mechanism is further classified as follows.

a) Full Checkpoint: It is a mechanism in which checkpoints to a process which is running are applied after a fixed interval of time which save the state of process to some media. If there is a failure of process during execution, it can be recovered from the last saved checkpoint state rather than restarting it. It should be considered that checkpoints should be applied after certain interval so that it does not cost much overhead and increase its execution time [17].

b) Incremental Checkpoint: This mechanism helps in reducing the checkpoint overhead by saving those pages in which there have been any change instead of saving the whole process [17-20].

3) Replication: It is a process of creating copies of similar data and then stored at different locations [5, 10-11, 14]. Replication can be further classified as follows.

a) Semi-active Replication: Each replica is provided input or state information, the primary replica as well as backup replica perform execution on the input provided. But the output is provided by the main replica. If the main replica fails, output is provided by backup replica. Fault tolerance manager (FTM) creates an equivalent replica for each replica which has failed and update its state. VMware's Fault Tolerance [21] falls under semi-active replication category.

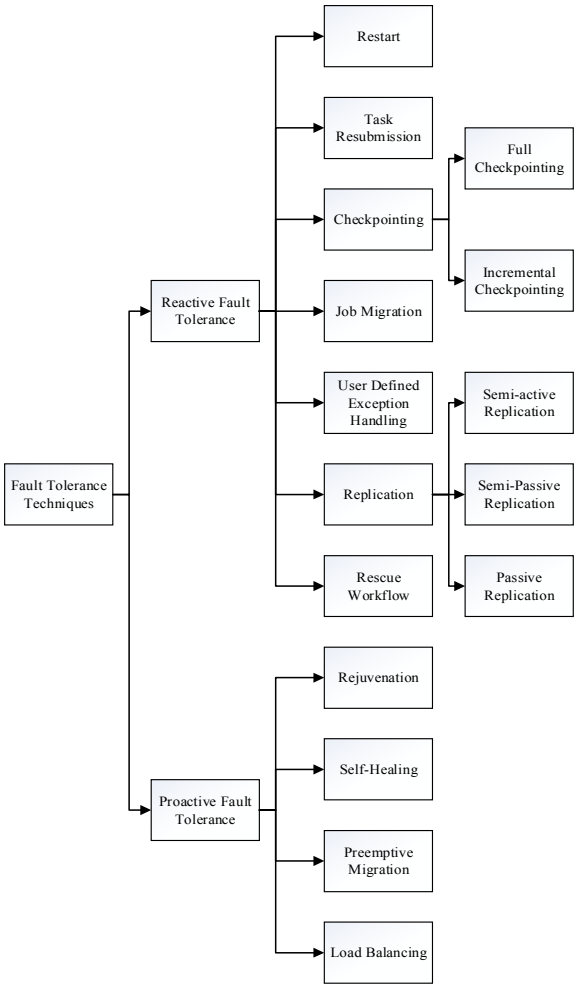

Fig. 5. Fault Tolerance Techniques

b) Semi-passive Replication: The main replica performs frequent checkpoints over state information and by saving input parameters in between each of the checkpoints. Replication is then done by moving the information of its state to all the backup replicas. The work of backup replics is to save the latest state obtained by primary replica they do not perform any execution. In case of failure of primary replica backup replica initiates and updates as primary replica with some loss of execution. Remus [22] is an example of semi-passive replication.

c) Passive Replication: A backup is created on which state information of a virtual machine instance is stored regularly. If there is a failure, fault tolerance manager recommissions another virtual machine instance which restores the last saved state. The backup can used for a particular application or it can be set up to share the state of several virtual machine instances, the virtual machine commissioning process can also be done with the help of priority value which is assigned to each virtual machine instance. VMware's High Availability solution [23] is an example of this technique.

4) Job Migration: During failure of a task the task which has failed can be migrated to another machine [11].

5) Task Resubmission: If there is a failed task detected in a system, the failed task is resubmitted to a new or same resource [11].

6) User Defined Exception Handling: For a particular treatment of a failure of task the user specifies a workflow in this process [11].

7) Rescue Workflow: Even if a task fails rescue workflow allows it to continue until or unless it is not possible to execute without considering the failed task [11].

B. Proactive Fault Tolerance

The proactive fault tolerance policies helps to avoid various faults by predicting them before they occur and replace the suspected component with new or non-faulty component before it actually occur [11]. Various techniques for proactive fault tolerance are discussed below.

1) Rejuvenation: It is issued before the system fails [16]. Assumptions are made as to when the system would fail if no measures were taken. Ideally, the rejuvenation interval would always end just before the system fails. A conservative choice of the rejuvenation interval will select short intervals. But rejuvenation comes with a cost of saving the operating environment and all processes, restarting the system and reinitializing the operating environment and all processes. If rejuvenation is performed too often the rejuvenation cost accumulates unnecessarily, if the system is rejuvenated at too long intervals it will often fail.

2) Self-Healing: Failures are handled automatically on those application if multiple instances of those application are run on different virtual machines [11].

3) Preemptive Migration: In this case a job which is executed is preempted, its state is then saved and then it is migrated to another system [11].

4) Load Balancing: Whenever the load (% utilization) of CPU and memory exceeds a certain limit. For example, if 75% utilization of CPU is considered as limit. The load from that CPU is transferred to other CPU which does not exceeds its limit.

Based on these techniques various fault tolerance tools are implemented as shown in Table I.

TABLE I. TOOLS FOR IMPLEMENTING EXISTING FAULT TOLERANCE TECHNIQUES

System	**Programming Framework**	**Fault Tolerance Techniques**
HAProxy[24]	Java	Job Migration, Self-Healing, Replication
SHelp[25]	SQL, Java	Checkpointing
Assure[26]	Java	Checkpointing, Self-Healing
Hadoop[27]	HTML, Java, CSS	Job Migration, Rescue Workflow, Replication
Amazon EC2[28]	Amazon Machine Image, Amazon Map	Replication, Task Resubmission

V. CONCLUSION

Fault tolerance is property of a system which allows it to provide required service with possibly degraded performance if there are one or more faults, or component failures. It plays a major role in providing availability and reliability of services to the user. This paper discuss various fault tolerance techniques which may cause failure of machines in cloud infrastructure.

REFERENCES

[1] Bo Yang, F. Tan and Y.S. Dai, "Performance Evaluation of Cloud Service Considering Fault Recovery," The Journal of Supercomputing, vol. 65, no. 1, pp. 426-444, 2013.

[2] Alain Tchana, Laurent Broto and Daniel Hagimont, "Approaches to Cloud Computing Fault Tolerance," in International Conference on Computer, Information and Telecommunication Systems (CITS), France, pp. 1-6, 2012.

[3] M. R. Lyu , Software Fault Tolerance, New York: Wiley, 1995.

[4] Algirdas Avizienis et al., "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33, 2004.

[5] Dawei Sun et al., "Building a High Serviceability Model by Checkpointing and Replication Strategy in Cloud Computing Environments," in 32nd International Conference on Distributed Computing Systems Workshops, China, pp. 578-587, 2012.

[6] Felix C. Gartner, "Fundamental of Fault Tolerance Distributed Computing in Asynchronous Environment," ACM Computing Surveys, vol. 31, no. 1, 1999.

[7] Lee Pike, Jeffrey Maddalon, Paul Miner, and Alfons Geser, "Abstractions for Fault-Tolerant Distributed System Verification," in Theorem Proving in Higher Order Logics (TPHOLs), volume 3223, pages 257–270, 2004.

[8] Arvind Kumar, Rama Shankar Yadav, Ran Vijay and Anjali Jain "Fault Tolerance in Real Time Distributed System," International Journal on Computer Science and Engineering (IJCSE), vol. 3, no. 2 , 2011.

[9] Yilei Zhang, Zibin Zhengand and Michael R. Lyu, "BFTCloud: A Byzantine Fault Tolerance Framework for Voluntary-Resource Cloud Computing," 4th International Conference on Cloud Computing, IEEE, 2011.

[10] Anju Bala and Inderveer Chana, "Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing," International Journal of Computer Science (IJCSI) Issues, vol. 9, no. 1, 2012.

[11] P. K. Patra, Harshpreet Singh and Gurpreet Singh, "Fault Tolerance Techniques and Comparative Implementation in Cloud Computing," International Journal of Computer Applications, vol. 64, no. 14, pp. 37-41, 2013.

[12] N. Chandrakala and P. Sivaprakasam, "Analysis of Fault Tolerance Approaches in Dynamic Cloud Computing," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 2, pp. 87-92, 2013.

[13] Anjali D. Meshram, A. S. Sambare and S. D. Zade , "Fault Tolerance Model for Reliable Cloud Computing," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 1, 2013.

[14] Ravi Jhawar and Vincenzo Piuri, "Fault Tolerance Management in IAAS Clouds," in IEEE First AESS European Conference on Satellite Telecommunications, Rome, pp. 1-6, 2012.

[15] Felix Salfner, Maren Lenk and Miroslow Malek, "A Survey of Online Prediction Methods," ACM Computing Surveys, vol. 22, no. 3, pp. 1-68, 2010.

[16] Katinka Wolter, Stochastic Models for Fault Tolerance Restart, Rejuvenation and Checkpointing, New York: Springer, 2010.

[17] N. Naksinehaboon et al., "High Performance Computing Systems with Various Checkpointing Schemes," International Journal of Computers, Communications & Controls., vol. 4, no. 4, pp. 386-400, 2009.

[18] D. I. Hunyadi and M. A. Musan, " Modelling of the Distributed Databases. A Viewpoint Mechanism of the MVDB Model's Methodology," International Journal of Computers, Communications and Controls., vol. 3, pp. 327-332, 2008.

[19] A. C. Palaniswamy, and P. A. Wilsey, "An Analytical Comparison of Periodic Checkpointing and Incremental State Saving," Proc. of the Seventh Workshop on Parallel and Distributed Simulation, California, pp. 127-134, 1993.

[20] N. H. Vaidya, "A Case for Two-Level Distributed Recovery Schemes," in Proceedings of the 1995 ACM SIGMET- RICS Joint International Conference on Measurement and Modeling of Computer Systems, pp. 64–73, 1995.

[21] VMware, "White paper: Protecting Mission-Critical Workloads with Vmware Fault Tolerance," 2009.

[22] B. Cully et al., "Remus: High Availability via Asynchronous Virtual Machine Replication," in Proc. of NSDI, San Francisco, pp. 161–174, 2008.

[23] VMware, "White paper: Vmware High Availability Concepts, Implementation and Best Practices," 2007.

[24] "HAProxy Configuration Manual" http://www.haproxy.org/download/1.5/doc/configuration.txt.

[25] Gang Chen et al., "SHelp: Automatic Selfhealing for Multiple Application Instances in a Virtual Machine Environment," IEEE International Conference on Cluster Computing, 2010.

[26] S. Sidiroglou et al., "ASSURE: Automatic Software Self-healing Using REscue points," Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'09), ACM Press, USA, pp.37-48, 2009.

[27] "MapReduce Tutorial" http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.

[28] "Amazon EC2" http://www.amazon.com/ec2/.