ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 3; Issue 3; May-June 2015; Page No. 116-120

A Survey on Fault Tolerance and its Techniques in Cloud Computing

Swati Pawar

Jayoti Vidyapeeth Women's University, Jaipur,

Rajasthan (India)

swati.pawar75@gmail.com

Abstract:
Cloud Computing is an auspicious computing platform for both commercial and non-commercial computation clients. It delivers services to the users transparently without their need to know the details of the underlying software and hardware. One of the crucial issues in cloud computing platforms and applications is fault tolerance. It is a major concern to guarantee availability and reliability of critical services as well as application execution. Fault tolerant system is one that has the ability to recover from a catastrophic failure without disrupting its operations. Fault tolerance techniques are used to minimize failure impact on the system and application execution by predicting failures and taking an appropriate action before failures actually occur.

Keywords: Cloud Computing, Fault Tolerance, Reactive, Proactive

1. INTRODUCTION

Cloud computing can be defined as a way of using computational resources such as storages, operating systems etc. which are located remotely and are provided as a service over internet[1]. Its advantages include low costs, high availability, scalability and elasticity. Although cloud computing has been widely adopted by the industry, still there are many research issues to be fully addressed. Fault tolerance is one of the key issues amongst all. The main benefits of implementing fault tolerance in cloud computing include failure recovery, improving reliability, lower cost and enhanced availability [2][3]. When a fault occurs, the fault tolerance techniques provide mechanisms to the software system to prevent system failure occurrence.

The rest of the paper is organized as follows. Section2 discusses about fault tolerance and its types in cloud computing. Section 3 presents techniques for fault tolerance in cloud computing. Section 4 finally concludes the paper and gives the future scope.

2. FAULT – TOLERANCE IN CLOUD COMPUTING

Fault-tolerance refers to correct and continuous operation of computing systems even in the presence of an unexpected problem or error. It is the use of redundancy to avoid failures due to faults. A fault tolerant system may be able to tolerate one or more fault types including- transient, permanent faults, design faults or operational faults. Each module has an ideal specified behavior and an observed actual behavior. A failure occurs of an error – a defect in the module and when the actual behavior deviates from the specified behavior[4]. The cause of the error is a fault[5]. The time between the occurrence of the error and the resulting failure is the error latency. This is illustrated in figure 1.

Figure 1: Generation path of failure

Fault tolerance is carried out by error processing which have two constituent phases – Effective error processing and Latent error processing. Effective error processing tries to correct the error after it becomes effective. Effective error processing may either recover from the error or mask the error. Latent error processing tries to detect and repair latent errors before they become effective. There are two forms of recovery, backward and forward recovery. Backward recovery returns to a previous correct state. The state is stored at regular intervals, and at restart time the last stored state is loaded and restarted from. In forward recovery, a new correct state is constructed, e.g. by re-sending a message or re-reading a disk page.

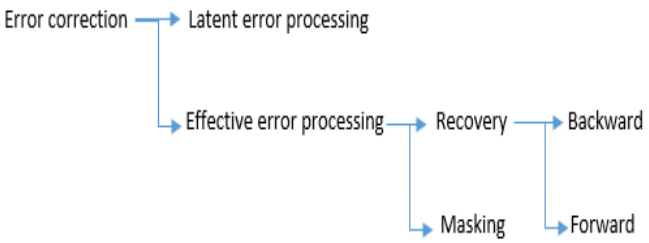

Figure 2: Taxonomy of error correction approaches

Corresponding author: Swati Pawar

2.1. Types of Fault-Tolerance

2.1.1. Hardware Fault-Tolerance

Hardware fault-tolerance is designed in systems to automatically recover from random faults occurring in hardware components and to prevent failures [6].

In this technique, components are separated into smaller modules that act to contain a fault. Each module is backed up with protective redundancy so that, if the module fails, other can assume its function[7]. There are two approaches of hardware fault recovery: 1) fault masking and 2) dynamic recovery.

- **Fault Masking**

Fault masking is a structural redundancy technique by which a number of identical modules are used to execute the same function. The output of the modules is compared to remove the errors in hardware computing. It uses a sort of voting protocol where if the main and backups don't give the same results, the flawed output is ignored. Triple modular redundancy (TMR) is a commonly used form of fault masking which can mask an error by executing three times and taking a majority vote. Hybrid redundancy is an extension of TMR in which the triplicated modules are backed up with additional spares, which are used to replace faulty modules – allowing more faults to be tolerated.

- **Dynamic Recovery**

A system with dynamic redundancy consists of several modules but with only one operating at a time. If a fault is detected in the operating module it is switched out and replaced by a spare. It requires consecutive actions of fault detection and fault recovery. In single computers special hardware is required along with software to do this, while in multicomputer the function is often managed by the other processors. Dynamic recovery is more space efficient than a voted system and it is better to be used in a resource-contained system or in a high-performance scalable system in which the active computing power should be maximized.

2.1.2. *Software Fault-Tolerance*

Software fault tolerance is used to describe the ability of software to detect and recover from a fault that is happening or has already happened when the software is running in order to provide service in accordance with the specification. Although the fault will not cause permanent faults, it may alter the processor's state, signal transfers or stored values on register, and affects a program's normal execution. It uses static and dynamic redundancy approaches similar to those used for hardware faults. An approach called design diversity combines hardware and software fault tolerance by implementing a fault-tolerant computer system using different hardware and software in redundant channels.

3. TECHNIQUES FOR FAULT TOLERANCE

Based on different fault tolerance techniques and strategies, we classify faults as follows:

- Reactive fault tolerance
- Proactive fault tolerance

3.1 Reactive fault tolerance

Reactive fault tolerance policies remove the fault after it occurs. This technique makes system more robust. There are various techniques which are based on policies of reactive fault tolerance like Checkpoint/Restart, Replication and Job Migration and so on[8].

- **Check pointing/Restart:** A checkpoint is a copy of the computer's memory that is periodically saved on disk along with the current register settings (last instruction executed, etc.) and any other status indicators [11]. When a failure occurs, the program can automatically detect the failure and restart from the checkpoint rather than from the beginning. It is an efficient task level fault tolerance technique for long running applications.
- **Replication:** Various task replicas are run on different resources, for the execution to succeed till the entire replicated task is not crashed. Unlike check pointing, the replication avoids task re-computation by executing several copies of the same task on more than one compute stations. It can be implemented using tools like HAProxy, Hadoop and AmazonEc2 etc.
- **Job Migration:** In this technique, during failure of any task, it can be migrated to another machine. This can be implemented by using HAProxy.
- **SGuard:** It is less disruptive to normal stream processing and makes more resources available. SGuard is based on rollback recovery. It checkpoints the state of stream processing nodes periodically and restarts failed nodes from their most recent checkpoints.
- **Task Resubmission:** Whenever a failed task is detected, it is resubmitted either to the same or to a different resource at runtime. It is the most widely used fault tolerance technique in current scientific workflow systems.
- **User defined exception handling**: In this technique user specifies the particular treatment of a task failure for workflows.
- **Rescue workflow:** This technique allows the workflow to continue even it becomes impossible to move forward without catering the failed task.

3.1.1 *Techniques used for Reactive Fault Tolerance*

- **FTM Architecture:**

FTM is a dedicated service layer placed between application layer and virtualization layer of Cloud Architecture. It is built to work on top of hypervisor, spanning all the nodes and traversing the abstraction layers of the Cloud to transparently tolerate failures among the processing nodes. A brief description on each element of FTM and their functionality is given below.

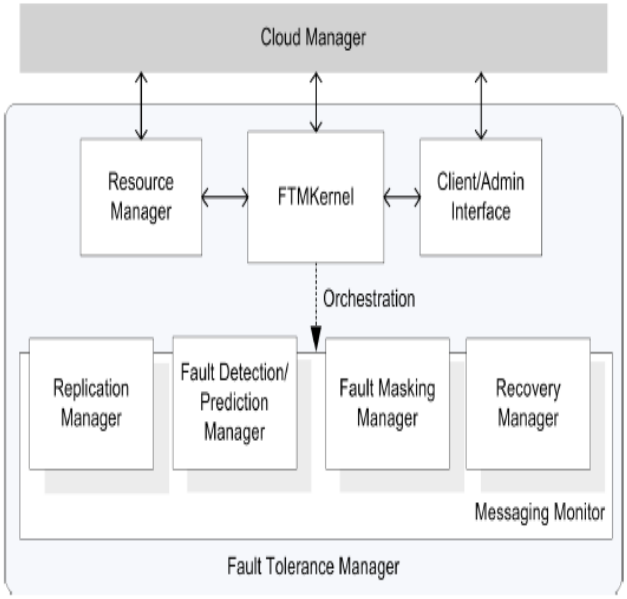

Figure 3: FTM architecture

In this architecture, the replication manager makes a redundant copy of the user's application so that it is available after failure happens. There are two types of replication methods: active and passive. In Active replication all copies of a task run along with the main instance concurrently in other virtual machines. Once a node fails, result from one of the active replicas could be used. In the case of passive replication, a backup is only executed when its corresponding task fails. In the Cloud computing environment, the set of VM instances can also be controlled by a single implementation of the replication service which is referred as a Replica Group. Fault Detection/Prediction Manager is used to detect replica failures[4]. It detects the faults immediately after their occurrence and send a notification about the faulty replica to the FTMKernel to invoke services from the Fault Masking Manager and Recovery Manager. There are two types of fault detection strategies: push model and pull model. In push model, Fault Detector check the health status by sending signals to various nodes. Whereas, in pull model method, each component in the system send signals to FD telling their health status. If no signal is obtained, FD considers that particular node is unhealthy and it reports to the master, so that, no more tasks are given to that particular node.

In Fault Masking Manager, FTM recover or replace the failed components in the background while the application's execution remains uninterrupted in another instance "replica". A collection of such algorithms that "mask" the occurrence of failures and prevent the faults from resulting into errors is included in this component. Recovery Manager continuously checks for the occurrence of faults and invoke the recovery service when exceptions happen. It includes all the mechanisms that resume error-prone nodes to a normal operational mode. FTMKernel chooses an appropriate messaging module while composing a fault tolerance algorithm such that an independent messaging facility is available for each instance. It makes the communication between any two components (and the replicas) reliable even in the presence of component, system or network failure [13]. Client/Admin Interface is used to obtain users' requirements and act as an interface between the end user and FTM. FTMKernel is the central computing component of FTM which manages all the reliability mechanisms present in the framework. It selects the web services from other components by contemplating the user's requirements.

- **MPI Architecture:**

MPI is a standardized and portable message-passing system for parallel computers. It provides a rich collection of point-to-point communication routines and collective operations for data movement, global computation, and synchronization. The "communicators" is the most advanced aspect of MPI. These makes it possible to isolate communication so that only those tasks which should take part in the message-passing can do so. It has a modular, layered architecture which separates the implementation of the high level protocols and functions from the low level mechanisms used for inter process communication and process management. The lower layer consists of a modular framework for collections of Systems Services Interfaces called SSI[9]. One such collection is the MPI Request between the MPI peer processes. Another is the checkpoint/restart (CR), which provides an interface to the back-end check pointing system that does the actual check pointing and restart functionality.

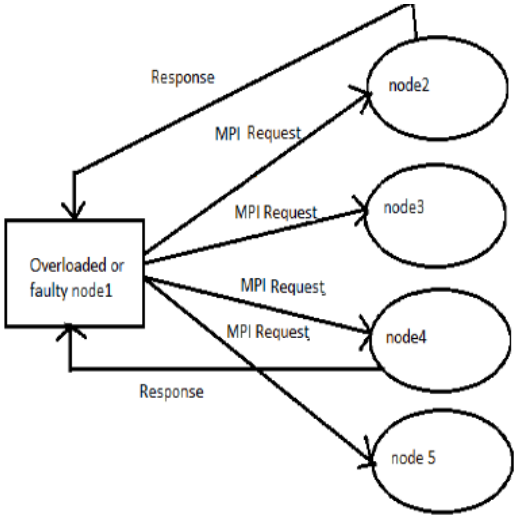

Figure 4: Message passing interface

3.2 Proactive fault tolerance

Proactive fault tolerance policies predict the faults, errors and failures in advance and proactively replace the suspected components with other working components. Some of the techniques which are based on these policies are Preemptive migration, Self-Healing etc.

- **Software Rejuvenation –** In this technique there are various rejuvenation intervals in an application and it restarted at every interval with a clean internal state. It designs the system for periodic reboots.
- **Proactive Fault Tolerance using Self-Healing -** When there are multiple instances of a single application running on different multiple virtual machines then by using self-healing the failure of different application instances can be handled automatically.
- **Proactive Fault Tolerance using preemptive Migration**: In Preemptive Migration, application is constantly monitored and analyzed and the parts of an application running on a computing node that is about to fail are migrated to a different node. By this the application is migrated to different node before actual fault occurs.

3.2.1 ***Techniques used for Proactive Fault Tolerance***

- **Map Reduce Fault Tolerance:**

The Map Reduce framework is a model that enables easy development of scalable parallel applications to process vast amounts of data on large clusters of commodity machines. It isolates the applications from the details of running a distributed program, such as issues on data distribution, scheduling and fault tolerance [10]. In this model, the computation takes a set of key/value pairs input and produces a set of key/value pairs as output. The user of the Map Reduce framework expresses the computation using two functions: Map and Reduce. The Map function takes an input pair and produces a set of intermediate key/value pairs. The Map Reduce framework groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function. The reduce function receives an intermediate key I with its set of values and merges them together. As shown in fig.5, in Map Reduce, input data is split into a number of blocks, each of which is processed by a map task. The intermediate data files produced by map tasks are shuffled to reduce tasks, which generate final data output. Map Reduce handles failures through re-execution. If a machine fails, Map Reduce reruns the failed tasks on other machines.

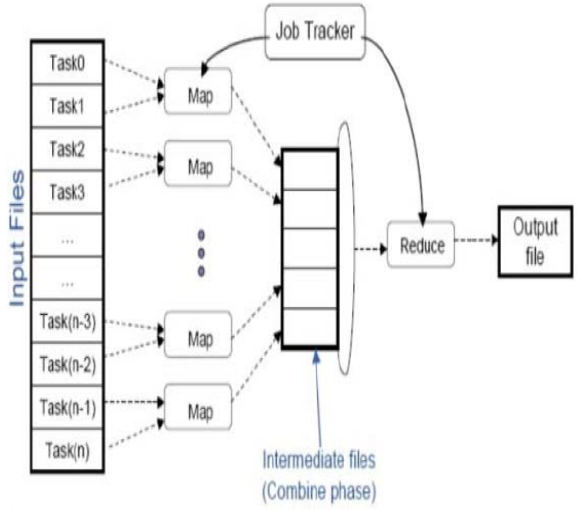

Figure 5: Map Reduce Architecture

4. CONCLUSION

Fault tolerance refers to correct and continuous operation even in the presence of faulty components. In this paper we described types of faults and various fault tolerance techniques. Several fault tolerance models are discussed. Presently, there is several mechanisms for fault tolerance but still there are number of challenges which needs to be considered. So, there is likelihood to overcome these challenges and try to make a solid model that may cover maximum fault tolerance aspect. We concluded that there is need of a more efficient and reliable technique that is also cheaper than the existing techniques.

5. REFERENCES

[1] Sun Microsystems, Inc. "Introduction to Cloud Computing Architecture" White Paper 1st Edition, June 2009

[2] AnjuBala, Inderveer Chana," Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012 ISSN (Online): 1694-0814 www.IJCSI.org

[3] Y.M. Teo, B.L. Luong, Y. Song, T. Nam, "Cost-Performance of Fault Tolerance in Cloud Computing, International Conference on Advanced Computing and Applications, (Special Issue of Journal of Science and Technology, Vol. 49(4A), pp. 61-73), Ho Chi Minh, Vietnam, October 19-21, 2011.

[4] Ravi Jhawar, Vincenzo Piuri, MarcoSantambrogioy," A Comprehensive Conceptual System-Levelapproach to Fault Tolerance in Cloud Computing,DOl10.1109 /Sys Con.2012.6189503.

[5] A. Christy Persya,Sr.Lecturer, T.R.Gopalakrishnan Nair," Fault tolerant real time systems, International Conference on Managing Next Generation Software Application (MNGSA-08), Coimbatore, 2008.

[6] http://en.wikipedia.org/wiki/Fault-tolerant_ computer_system

[7] http://www.cs.ucla.edu/~rennels/article98.pdf

[8] Amritpal Singh, Supriya Kinger, "An Efficient Fault Tolerance Mechanism Based on Moving Averages Algorithm" © 2013, IJARCSSE, ISSN: 2277 128X.

[9] EkypeOkorafor, "A Fault-tolerant High Performance Cloud strategy for Scientific Computing" ©2011 IEEE, DOI10.1109/IPDPS.2011.306.

[10] Qin Zheng, "Improving Map Reduce Fault Tolerance in the Cloud" ©2010 IEEE.

[11] Golam Moktader Nayeem, Mohammad Jahangir Alam,"Analysis of Different Software Fault ToleranceTechniques", 2006.

[12] Geoffroy Vallee, Kulathep Charoenpornwattana, Christian Engelmann, Anand Tikotekar, Stephen L.Scott," A Framework for Proactive Fault Tolerance".

[13] G. Vallee, K. Charoenpornwattana, C. Engelmann, A. Tikotekar,C. Leangsuksun, T. Naughton, and S. L. Scott, "A Framework for Proactive Fault Tolerance," in Proceedings of the 2008 Third International Conference on Availability, Reliability and Security. Washington, DC, USA: IEEE Computer Society, 2008, pp. 659–664.