Inspection of Fault Tolerance in Cloud Environment

Deepanshu Jain(✉), Nabeel Zaidi, Raghav Bansal, Praveen Kumar, and Tanupriya Choudhury

Amity University Uttar Pradesh, Noida, India
{deepanshujain002, raghavbansal95}@gmail.com, nabeelzaidi@ymail.com, {pkumar3, tchoudhury}@amity.edu

Abstract. Cloud environment is a set of various types of software and hardware that are connected with each other and works collectively to provide various services to the user as an online utility. It basically is the efficient use of hardware and software to work coherently to deliver services. Through the use of cloud computing, users are able to access files from almost anywhere or any device that have access to Internet. We are already familiar with the market of cloud computing, and how everyone is shifting to cloud because of the benefits it provides. But many of them are very less aware about the situations when there comes a failure. The task of facing the failure is not limited to the cloud providers but also to the customers. They must know what can be done when such a situation arises. Fault tolerance is basically a property that makes the system to work properly even though there is a failure. In order to be more robust and dependable, failure should be handled effectively. This paper deals with the Inspection of various fault tolerance technologies that are available. There is no existing algorithm that considers reliability and availability in fault tolerance as well. We tried to consider these things when discussing about fault tolerance. Furthermore, a brief analysis of the already proposed FTC model with some new functionally is also presented.

Keywords: Cloud · Replicating · Scheduling · Cluster

1 Introduction

In order to provide a network access to a shared pool of resources which can be enabled/disabled on demand and is universal as well as convenient, a special kind of framework is used which is known as cloud computing. Using cloud computing, we provide access to the least amount of management and very less or no interaction with the service provider [1]. One of the basic processes of using cloud as service is to be able to provide resource processing for which scheduling is necessary. Every cloud application is designed keeping business processes in mind and includes a set of abstract functions or services. And to be able to process the tasks, the system needs to allocate both the resources as well as the tasks that come to the resource. And to ensure the Quality of Service also known as QoS, there must be a Service Legal Agreement

(SLAs) in place [2]. The National Institute of Standards and Technology states that the Cloud Model consists many service and deployment models along with some key essential characteristics [1].

1.1 Essential Characteristics

1. On-demand self-service—An end user can automatically get new services granted to it, may it be server or storage and that too without any interaction with the service provider [3].
2. Broad network access—The services provided can be used across many platforms ranging from thin platforms such as mobile phone to thicker platforms such as workstations and laptops.
3. Resource pooling—All the resources provided by the service provider are pooled together to serve multiple users at the same time by dynamically allocating resources according to the demand of the consumer who has no knowledge about the exact location of the end user unless it is at a higher abstraction [4] level.
4. Rapid elasticity—Services and resources provided to the users can be granted and released automatically in order to compete with the scaling may it be outward or inward. From the customer's point of view, the amount of resources may seem to be unlimited which can be granted to them at any point of time.
5. Measured service—Cloud system has a metering capability at a certain level to abstraction which can be used by them to monitor the resources, optimize the resources, and controlling them. This process is transparent to both the service provider as well as the end user.

1.2 Service Models

1. Software as a Service (SaaS)—The consumer is given the capability to use the service provider's application which is running on the existing cloud infrastructure. The client can get access to the application from various devices using Web browser as a thin interface or any other program interface. The infrastructure of the cloud is not controlled by the end user with the exception any configuration setting, that is, provided in the application itself.
2. Platform as a Service (PaaS)—The consumer is given the ability to deploy the applications that the consumer has created using tools provided by the service provider or acquired onto the cloud. The consumer has control only over the application, its configuration setting, and the environment over which the application is hosted and has no control over the cloud infrastructure.
3. Infrastructure as a Service (IaaS)—The consumer is provided with many capabilities such as storage, provisional processing, and networks among other resources using which the consumer is given the ability to run and deploy many software including operating systems. The only thing consumer can control are the resources that are provided to them and the applications they deploy, and they have no control over the infrastructure of the cloud.

1.3 Deployment Models

1. Private cloud—This type of cloud infrastructure may be owned by a particular organization and is used exclusively by the members of that organization only. It can be under the ownership of the organization or a third party or a combination of both.
2. Community cloud—This type of cloud infrastructure is used by the members of a specific community from an organization that have similar concerns. It also maybe be under the ownership of the organization to which the community belongs or a third party or maybe a combination of both. This type of infrastructure may exist either on premises or off premises.
3. Public cloud—This type of cloud infrastructure is not made for any exclusive use and is open to all. It can be under the ownership of a business, government, or any academic organization that are also responsible for its maintenance. It always exists on the premises of the organization providing the service.
4. Hybrid cloud—As the name suggests, this type of cloud infrastructure is a combination of two or more types in which the unique attributes is that each infrastructure exist, and they are bounded together by some standardized technology which allows data and application portability.

Since cloud computing allows its users to have access to resources and services for as long as they need them and that too without too many interactions, it is becoming perhaps one of the fastest growing grid technology [1, 5]. Cloud computing focuses on sharing of information among its various customers by putting up the same information on the grid of nodes [6]. For the management of resources, the following aspects must be kept in mind:

1. Infinite resources that must be made available to the consumer on their demand.
2. No commitment from the users in advance.
3. Last but certainly not the least would be the demand of high-end computing resources and that too for very short duration of time.

Implementation of fault tolerance is very important from the perspective of the service provider since it rescues the lost, improves the performance, and can also be used in failure recovery, and in order to achieve it, some mechanisms such as redundancy and replication can be used [2]. In real-time computing applications, adding a cloud infrastructure does not only mean increasing the chances of error, but it also increases the cost as the resources required to keep the replicated data increase [5]. And realizing the power of cloud, some cloud providers have even started to give real-time cloud support since the power of cloud can prove to a great added benefit to real-time applications [1]. Cloud computing infrastructure usually comprises of interconnected data centers and infinite resources which are provided to the consumer as a part of an on-demand service [7]. In order to get reliable software, fault tolerance is a necessary demand but it proves to be great difficulty to design an integrated solution since many of the user's applications are deployed on the cloud infrastructure only. The major cause of this difficulty is the complexity of the system and multiple abstraction layers due to which only limited data is available [7].

2 Resource Manager

The service provider has to keep a consistent sight of all the systems so that the resources can be assigned to each client request systematically. Hence, a database of catalogue which consists of the current state of resources is maintained by the resource manager whose main function is to observe the current state of all virtual as well as physical resources. The resource manager, therefore, keeps catalogue of each and every machine in its database along with other information of the system such as its serial number, speed of the processor, and the date the resource was issued. [7]. Given below is Fig. 1 of resource graph G(N, E) of the cloud infrastructure [8, 9] consisting of two clumps of three nodes each which are connected via network switch. In the graph, processing nodes n∈N are represented by the vertices, and to represent virtual connection between two nodes, edges of the graph are used, i.e., e∈E. Here, each node keeps the information about the virtual machine that is kept at that particular node in the vertex. Each edge and node can be further categorized into three classes which are: working(W), completely faulty(F), and partially faulty(F). Each node is marked with a particular class depending on its current state. If it is an ordinary state, it will mark W. If a failure has occurred such that the node cannot be recovered back to a normal state, it is marked as F, and the node which is currently not in use is marked as P [7].

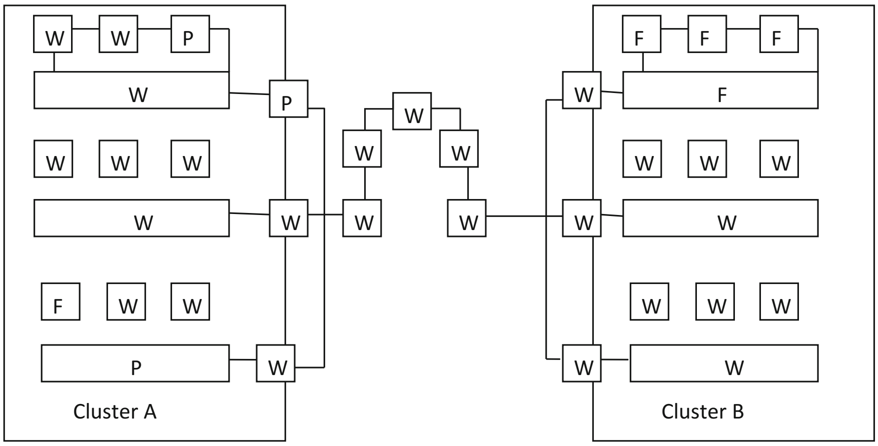

Fig. 1. Graph generated by the resource manager

Resource manager plays a very vital role in providing stability to the cost of the resource and toward the performance of the fault tolerance method that is used by the service provider [7].

3 Fault Tolerance in Cloud Computing

3.1 Basic Notation to Fault Tolerance

When a client needs fault tolerance services, he/she is enlisted by the service provider. Then, based on the requirements of the client, the service provider [10] creates a solution. While creating the solution, the balance between the following aspects must be kept in mind:

1. Fault Model—It defines the maximum capacity of the solution to handle faults and loss. In order to specify this aspect, the ability of the system to handle protocols of failure detection and the technology used is of grave importance.
2. Resource utilization—It defines the amount of resources that will be utilized in order to understand the fault. This feature is inbuilt in the system along with harsh level of failure detection and recovery.
3. Performance—The performance of any failure tolerance method is defined by the effect of the solution on the overall quality of service both at the time of failure and at times when no failure is there.
4. Redundancy—This technique is the most commonly used way of dealing with failures in a system. A failure tolerance model based on this technique replicates the components of the analytic system with the help of other resources so that these duplicated items can be used at the time of failure [7].

3.2 Fault Tolerance and Reliability

Fault tolerance and the overall reliability of the system are perhaps two of the most important aspects of cloud computing. To be able to provide consumers with the right solutions even with the presence of faults is of utmost importance to the consumers and the service providers as well. And as most of the service providers are moving toward providing a real-time experience, hence the demand of fault tolerance techniques for real-time systems is increasing drastically. But in most of the services that provide real-time experience, the processing is done on systems that remote and are on the cloud due to which the probability of error increases as consumers have very little control over those computing nodes. Hence, fault tolerance solutions are provided to the consumers so that such errors can be predicted before they actually take place. Moreover, even the reliability of virtual machine does not remain constant; it changes after each and every computing cycle [5].

Fault tolerance is done in two phases; first phase is “Effective Error Processing” in which the system is brought back to a state before the error took place, i.e., the dormant state, and the second is “Latent Error Processing.” Real-time systems are characterized by two main features which separate them from any other kind of system, and these are timeliness and fault tolerance. Timeliness means that every task must complete its execution in the given time period, and fault tolerance means that the system must continue to work even if any fault arises [1]. Hence, with the growing popularity of cloud computing, service providers have come with a new design approach in which fault tolerance mechanism is provided as a module-independent service so that this service can be provided to all the users and by each and every module transparently [7].

In order to achieve fault tolerance, a set of design techniques and algorithms are applied to increase the overall dependability of the system. With new upcoming technologies, new applications arrive, and hence, new fault tolerance solutions have to be introduced as well. Earlier specific hardware and software used to be made in order to provide fault-tolerant execution of tasks but the new microprocessor chips are highly complex, and all the hardware and software are made according to pre-defined norms which are economically feasible as well. Hence, many new techniques have come up to the surface in the field of fault tolerance such as using the existing technique with RAID disks where all the information is divided among many disks, and this improves the bandwidth. Moreover, an extra disk stores the encoded information to restore the data in case of system failure. Fault tolerance techniques are also being used in parallel computers in order to detect faults and errors. Fault tolerance techniques are becoming

Table 1. Comparison of fault tolerance strategy [1]

S. No.	Strategy	Fault-tolerant technique	Programming framework	Environment	Faults detected
1.	Nicolae and Cappello (2011)	Disk-based Checkpoint	MPI	IaaS cloud	Node/network failure
2.	Hakkarinen and Chen (2013)	Diskless-based Checkpoint	NA	HPC	Process/application failure
3.	Kwak and Yang (2012)	Checkpoint	Probability analytic framework	Real-time systems	Process failures
4.	Goiri et al. (2010)	Checkpoint	Java	Virtual machine	Node failure
5.	Malik et al. (2011)	FTRT (Adaptive)	–	Real time	–
6.	Sun et al. (2013)	DAFT (Adaptive)	Java	Large-scale cloud	Works on historical failure rate
7.	Cogo et al. (2013)	FITCH (Adaptive)	Java	Large-scale cloud	–
8.	Zhang et al. (2011)	BFT Cloud (Adaptive)	Java	Voluntary resource cloud	Byzantine problems
9.	Zhao et al. (2010)	LLFT (Adaptive)	C++	Middleware	Replication faults
10.	Ko et al. (2010)	IFT(Adaptive)	Hadoop	Hadoop	Intermediate data faults
11.	Zheng (2010)	MFTLL (Adaptive)	MapReduce	MapReduce	Replication faults, stragglers detection
12.	Pannu et al. (2012)	AAD (Adaptive)	–	Local cloud	Discovers future failures

more famous day by day especially in sub-micron VLSI in order to solve major problems such as noise and improving the overall yield of the system by increasing its ability to process even with faults.

We have compared the various fault tolerance techniques which are quite famous ones as shown in Table 1.

4 Related Works

The model is already proposed and is called fault tolerance in cloud computing (FTC) [1]. We are going to analyze it more deeply and give more functioning to the model. This model tolerates the faults based on the reliability each node has. Each node to be executed is taken to be a real-time application. The model is shown in the figure given below. Here, we have "N" computing nodes or virtual machines each of which is running a different algorithm. Further, we perform an acceptance test (AT) whose result is then forwarded to the adjudication node to take a decision regarding it [2, 5]. Here, two distinct nodes are displayed, one of which contains some virtual machines on the cloud infrastructure and running different algorithms to handle real-time applications. The proposed algorithm then supplies the result again for an acceptance test to see whether it is logically valid or not. The test modules are similar to each other in every sense. If the results are valid, only then they are passed to the time checker module; otherwise, the AT modules send an exception signal reflecting the reason. The proposed scheme just not provides forward recovery but sometimes also can provide backward recovery. The other node is the adjudication node which comprises of three separate components: First, the Time Checker [TC] Module, it checks the timing of the results produced as it contains a timer that records the timestamp at which each result was produced [11]. Second is the Reliability assessor (RA) module which is used to check the reliability of each computing node. The final component is the Decision mechanism (DM) module which selects the best output based on their reliabilities.

As a cloud infrastructure consists of more than one grid, it becomes hard to tackle various security issues like confidentiality and integrity [6]. In [12], a fault-tolerant

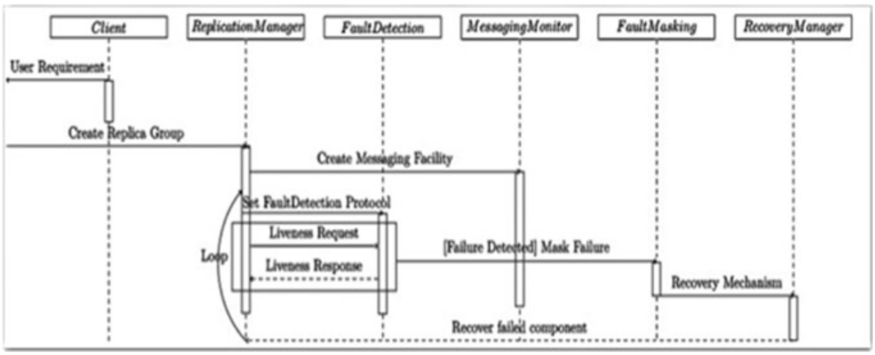

Fig. 2. FTM Kernel [1]

middleware is being proposed by the authors that can use replication to handle the faults in real-time cloud applications. In [7], in order to build a fault-tolerant protocol, we propose to use micro-protocols and use them in hierarchical order to form a system. In [6], to develop a fault-tolerant framework that is proactive as well, we propose to use a modular approach that can incorporate a requirement-specific strategy as well (Fig. 2).

The model already proposed can be used in real-time applications based on cloud infrastructure to handle the faults since the model can tolerate faults to a high extent. It has high reliability which can be dynamically configured. Moreover, we tried to introduce an approach that can be used in order to recognize generic fault-tolerant mechanisms as independent modules, the properties of each mechanism have been validated, and the user requirements have been matched with the available fault-tolerant mechanisms.

5 Conclusion and Future Work

A great analysis and design techniques are applied to create improved systems in fault tolerance. Almost every day, a new technology and applications are being developed so there is a need for new approaches to fault tolerance. Previously, it was quite easy to craft a specific hardware and software for a solution, but as technology is getting more advanced, it is getting complex too to apply a solution. Thus, there is a great deal of current research focusing on implementing fault tolerance. In this paper, we have successfully inspected the fault tolerance technology. We have discussed it with reliability and availability, and also, we have added new functionality to FTC-proposed model.

In future, we look forward to implement the proposed framework in order to measure the strength of the fault tolerance approach. Work is also proposed to create a new module called Resource Awareness Module or RAM that can help the cloud service provider's scheduler to schedule decision that is based on the characteristics of the infrastructure of the cloud system.

References

1. Dilip Kr Baruah, Lakshmi P. Saikia, "A Review on Fault Tolerance Techniques and Algorithms in Cloud Computing Environment", in International Journal of Advanced Research in Computer Science and Software Engineering Volume 5, Issue 5, May 2015.
2. Michael K. Reiter and Avishai Wool, "Probabilistic Quorum Systems", Information and Computation 170, 184–206 (2001).
3. Apolinar González-Potes, Walter A. Mata-López, Vrani Ibarra-Junquera, Alberto M. Ochoa-Brust, Diego Martínez-Castro, Alfons Crespo, "Distributed multi-agent architecture for real-time wireless control networks of multiple plants".
4. Zohaib A. Faridi, S. Rawat "Analysis and proposal of a novel Approach to collision detection and avoidance between moving objects using Artificial Intelligence" 5th Fifth International Conference on System Modelling & Advancement in Research Trends in TMU, Moradabad (UP) 25–27 Nov. 2016.

5. Alain Tchana, Laurent Broto, Daniel Hagimont, "FaultTolerant ApproachesinCloudComputingInfrastructures", in ICAS 2012: The Eighth International Conference on Autonomic and Autonomous Systems.
6. John D. Slingwine, Paul E. McKenney, "Apparatus and method for achieving reduced overhead mutual exclusion and maintaining coherency in a multiprocessor system utilizing execution history and thread monitoring".
7. Hagit Attiya[2], Alla Gorbach[3], Shlomo Moran[4], "Computing in Totally Anonymous Asynchronous Shared Memory Systems", Information and Computation Volume 173, Issue 2.
8. Praveen Kumar, Dr. Vijay S. Rathore "Improvising and Optimizing resource utilization in Big Data Processing" in the proceeding of 5th International Con-ference on Soft Computing for Problem Solving (SocProS 2015) organised by IIT Roorkee, INDIA (Published in Springer), Dec 18–20, 2015. PP 586–589.
9. Seema Rawat, Praveen Kumar, Geetika, "Implementation of the principle of jamming for Hulk Gripper remotely controlled by Raspberry Pi" in the pro-ceeding of 5th International Conference on Soft Computing for Problem Solving (SocProS 2015) organised by IIT Roorkee, INDIA, Dec 18–20, 2015. PP 199–208.
10. Sheril Yadav "Analysis and Implementation of Business Intelligence Software for Report Bursting" International Conference in Smart Computing & Informatics (SCI-2017) held in ANITS, Visakhapatnam March 2017.
11. Michael Wei, Amy Tai, Chris Rossbach Ittai Abraham, "Silver: Ascalable, distributed, multi-versioning, Alwaysgrowing(Ag) FileSystem".
12. HyoJong Lee, Shwetha Niddodi, David Bakken, "Decentralized voltage stability monitoring and control in the smart grid using distributed computing architecture" published in Industry Applications Society Annual Meeting, 2016 IEEE.