ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

*International Advanced Research Journal in Science, Engineering and Technology*
*Vol. 2, Issue 2, February 2015*

# Efficient Fault Tolerant Strategies in Cloud Computing Systems

**Siddharth Shyamsunder[1], Kunal Varkhede[2], Shirish Bhuruk[3], Rasika Vibhute[4], Prof R.P. Karande[5]**

Student, Computer Engineering, NBN Sinhgad School of Engineering, Pune, India[1,2,3,4]

Assistant Professor, Computer Engineering, NBN Sinhgad School of Engineering, Pune, India[5]

**Abstract:** To handle the large amount of incoming data, cloud computing has emerged as a very efficient technology. Cloud computing makes use of strategies that involve the use of an on-demand service of providing multiple virtual servers. As cloud computing serves many techniques for handling big data, a need arises to detect the faults that occur and to manage those faults in such a way as to keep the fault tolerant measures, abstract from the user. Here we are going to provide various fault tolerance techniques by making use of a service layer. The responsibility that is given to the service layer will be to detect the faults that occur at the server and to accurately find a solution keeping the internal details hidden from the user.

**Keywords**: Introduction, Basic concepts, Related work in the area of cloud computing fault tolerance, functionality of our proposed system, future scope and conclusion, references

## I. INTRODUCTION

Cloud computing deals with making use of an on-demand service to handle the huge amounts of incoming data. It proposes a system of making use of operational expenditure (OPEX), rather than the traditional use of capital expenditure (CAPEX). Hence a pool of servers is provided virtually in order to handle separate amount of data. The number of servers is extensible and wholly depends upon the rate of incoming data. While cloud computing proves to have many benefits, this technology has also become a victim to a number of risk factors, especially since the faults like that of server crashes, network congestion or server overload, that are found between the server and its data centres are common faults, nevertheless faults that are complex to handle in a system level perspective.

There is a means to handle these faults by making use of efficient fault tolerant strategies. In doing so Care must be taken to ensure that the software performance doesn't deteriorate and the transmission of data is highly reliable.

The strategy of detecting errors is enacted by comparing system specifications to that of the client's requirements. There are some other parameters that must be considered during this comparison, such as that of the complexity of the system, abstraction of the details from the user and linkage of the faulty server with that of every other server.

If a case of a faulty component arises in a data center, we make use of an efficient strategy of job migration, wherein we switch the faulty server with that of another server with proper working conditions. Here we are only going to make use of those servers, who have one of the below three requirements in common-

1. The new server must meet that of the user's specification.
2. There must be a proper linkage between the faulty server and the client's server.
3. The new server that replaces the faulty server must be in the list of most frequently used in terms of success rate.

This paper will further contain the following sections. Section 2 will focus on basic concepts, Section 3 will deal with the related work in the fault tolerance area of cloud computing. Section 4 will focus more upon the functionality of our system as a whole.

## II. BASIC CONCEPTS

Faults: In general to deal with fault tolerance, it is necessary to understand the concept of faults, errors and failures. A failure is that obstacle that renders us unable to reach a proper solution. These failures are mainly caused due to errors. A fault on the other hand is that error which is created due to incompatibility of meeting with the system's requirements. One can consider the flow as such

$$\ldots\text{Fault} \longrightarrow \text{Error} \longrightarrow \text{Failure}\ldots$$

Fault Tolerance: Fault tolerance is nothing but the ability to handle the faults and remove failures in such a way that the software will perform its functions unaware that any methods have been taken to handle the faults.

Fault model: The fault model deals with the measures that are taken in order to increase the rate at which the faults are efficiently solved. This is carried out by keeping in mind the rate of fault handling and the reliability at which the system provides the right response to the client's request. Keeping this in mind we have to actively judge precautionary measures so as not to reduce the performance of our system.

Some of the active stakeholders that are addressed in our system include:

1. Fault Detector: The fault detector plays the active role in the detection of faults that occur in the system. Their major role is to scan the system's server, and to send to the FTM the type of fault generated through means of protocols, so that the FTM can handle the system efficiently.

2. Resource Manager: The resource manager provides the necessary information regarding the system status and specifications of the different clusters.

3. Replication Manager: The replication manager handles the concept of check pointing. This concept underlies the fact that a number of backups are created, and if a case occurs such that the fault isn't solved, then the data is restored from the backup replicas, else the backups are updated with the new processed data.

4. Fault Tolerance Manager: The fault tolerance manager is the active head that manages the entire Fault tolerance system.

5. Recovery Manager: The recovery manager helps the FTM in recovering the backups from the replication manager.

## III. RELATED WORK

The server components in a Cloud computing environment are subject to a huge array of failures, each affecting user's applications. For example, a crash in the pair of aggregate switches may result in the damage of the servers that are linked to a particular cluster. Hence here, the fault might be affecting only one cluster, and the other clusters might work in perfect order. To handle this kind of error, a replica is made on the different clusters, so as to ensure proper recovery in case of an unavoidable fault. Our focus is on associating of a replica with a unique id and placing them on separate clusters through hashing techniques.

Cloud computing as such is divided into three independent layers in terms of service, i.e. Information as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The administrator manages 2 levels i.e. the hosting server and the client who is requesting for service.

In our proposed system the client who requests for fault tolerance support is indirectly managed by the FTM such that the FTM compares system's specification with that of the user's request.

Timeliness is a big issue, especially when we consider that of real time cloud systems. Real time cloud systems are rapidly emerging.

## IV. SYSTEM FUNCTIONS

The product functions to handle the ever increasing faults that are becoming a major threat in the case of performance, reliability and various other factors

We are going to make use of a fault detector and a replication manager. When the client sends his request to the server, the Replication Manager automatically creates a number of backups to client information. Once the replicas have been successfully created, the fault detector will compare application specifications with that of the server specifications, if any criteria are observed that may result in a server fault or overload (in case of improper memory load to the server system). In such a case, A signal is sent to a pool of clusters, where the success rates as well as the server system status are recorded. Individual speeds to the different servers are noted and a proper algorithm is generated to select the precise resource to handle the situation. If in any such case a particular resource cannot be allocated, the previously stored data is extracted from the replication manager and switched to the server that is closest and has the highest performance rate in terms of the success ratio. If a previous fault has been resolved, the updated transaction can be duly be handled by the recovery manager and the information is then sent to server to respond to client request.The Resource manager handles the clusters through means of a nodal system, very much like a nodal graph. The nodal graph consists of a set of nodes, connected in a random network topology. Here we are going to consider those nodes that actively meet the client's requirements. We are also going to use a ranking system that connects the faulty server with the most frequently used server.

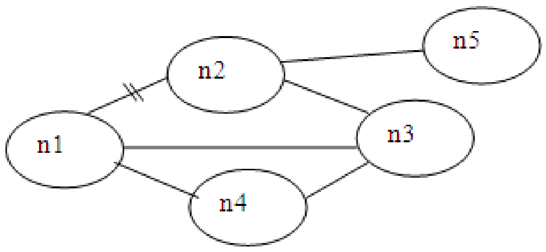

Fig.1. Example of a resource graph

The above graph shows the linkage between the different nodes that are handled by the resource manager. The nodes are considered in set form as N = {n1, n2,..,nN}. The edge or the path between n1 and n2 has two lines horizontally passing through it. This means that although there is a path between n1 and n2, there is a faulty linkage between them. Hence this path cannot be selected.

Once the linkages are properly set, the next step will be to find the most popular server or rather the server that has the most number of frequent visits. This can be achieved by assigning a binary code to each node through Huffman's coding and finding the node with shortest number of bits.

Once that is gathered, the comparison is carried out between the ft_units and user requirements. If matched, the response is sent to client else, the backup is recovered.

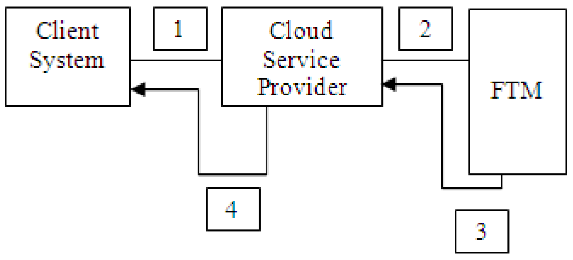

Fig.2. Abstract Overview Of Fault Tolerant System.

The overall Abstract view of the system can be seen as in three blocks. The first block has the client that sends a request to the server. on detection of a fault, the FTM takes control of the server's request, which is then subjected to many processes to handle the fault. Once the fault is properly handled, the response is sent to the server, which in turn sends the client's response.

1. Client sends request to server.
2. Server sends request to FTM to handle fault.
3. FTM handles the fault and sends a response back to the server.
4. Server sends the response to the client.

## V. CONCLUSION AND FUTURE SCOPE

*A.Conclusion*

Cloud computing is a highly rapid technology to handle the huge amount of incoming data efficiently.

This might further lead to the increased number of faults pertaining the system.

Hence it is necessary to handle this problem, it is necessary to provide efficient fault tolerance techniques.

The technique used by our system is nothing but an efficient way of job migration and check pointing of resources to handle certain faults like server crash and network congestion.

Also proper congestion control techniques are carried out to handle congestion control.

*B.Future Scope*

Our proposed approach is dynamic in nature. In other words, we are simply extending the features that are present in existing fault tolerance systems.

Our approach focuses on providing benefits to all our stakeholders by providing improved performance and faster fault tolerance management.

Some of the other less known future enhancements also include proper measures to be taken in order to improve the security and usability of the system. Compatibility of the fault tolerance system to that of the ever increasing as well as the change in demands of the user's requirements can further be enhanced. This means that the system can be made to be maintained to handle the fast pace of the technology.

Besides this further performance measures can be provided, adhering to the cloud standard as well as providing more efficient fault tolerance techniques to handle the increasing number of bugs in Fault Tolerance Management.

## ACKNOWLEDGMENT

We would like to give our sincerest thanks to **Dr. R.S.Prasad**, Head of Department computer Engineering and our guide **Prof.R.P.Karande** for giving us valuable support and for presenting us with opportunities to present our ideas.

## REFERENCES

[1] Amazon Elastic Compute Cloud [Online]. Available: *http://aws.amazon.com/ec2/*

[2] Eucalyptus Systems [Online]. Available: *http://www.eucalyptus.com/*

[3] K. V. Vishwanath and N. Nagappan, "*Characterizing cloud computing hardware reliability*," in Proc. 1st ACM Symp. Cloud Comput., 2010, pp. 193–204.

[4] V. Piuri, "Design of fault-tolerant distributed control systems," *IEEE Trans. Instrum. Meas.*, vol. 43, no. 2, pp. 257–264, 1994.

[5] L. L. Pullum, *Software Fault Tolerance Techniques and Implementation*. Norwood, MA: Artech House, 2001

[6] Vincenzo Piuri,"Fault Tolerance Management In Cloud Computing- A System Level Perspective " IEEE TransInstrum..2013