IOP Conf. Series: Materials Science and Engineering **263** (2017) 042060 doi:10.1088/1757-899X/263/4/042060

Advanced cloud fault tolerance system

K Sumangali and Niketa Benny

VIT University, Vellore-632014, Tamil Nadu, India.
E-mail : ksumangali@vit.ac.in

Abstract. Cloud computing has become a prevalent on-demand service on the internet to store, manage and process data. A pitfall that accompanies cloud computing is the failures that can be encountered in the cloud. To overcome these failures, we require a fault tolerance mechanism to abstract faults from users. We have proposed a fault tolerant architecture, which is a combination of proactive and reactive fault tolerance. This architecture essentially increases the reliability and the availability of the cloud. In the future, we would like to compare evaluations of our proposed architecture with existing architectures and further improve it.

1. Introduction

Cloud computing is a model for facilitating ubiquitous, convenient, on-demand network access to a pool of resources such as servers, applications, services and networks that are shared and can be allotted or removed with minimal effort from the management or service provider[1]. The profits of cloud computing are massive but this innovative paradigm has completely changed the dimension of risks on client applications, the reason behind this is all the failures that occur in data centres which are not part of the client's organization such as server overload, network congestion and hardware faults. These failures impose high consequences on the client's application that is deployed in virtual machines and hence, it is a necessity to address user's reliability and availability concerns. Fig. 1 depicts a rudimentary cloud computing architecture, where the end users access the client's application though virtual machines and the virtual machines are hosted using the cloud provider's resources.

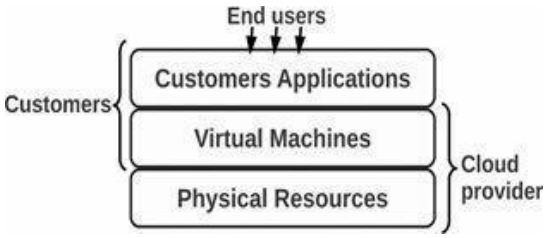

Fig. 1. Cloud computing architecture

Fault tolerance techniques are generally used to predict these failures and take the necessary action before a client's application actually gets affected. The two important concerns for the requisites of fault tolerance is reliability and availability. Reliability is the ability of a system to perform in accordance with its specifications. Ideally, a reliable product is completely free of technical errors and availability is the ratio of time a system is actually functional to the total time it is required or expected to function.

Fault tolerance techniques on cloud computing devices can be classified into two major categories: reactive fault tolerance techniques and, proactive fault tolerance techniques. In reactive fault tolerance, it tends to minimize the impact of failures on the client's application in case of failure in one of the components. Proactive fault tolerance techniques tend to predict failures and take precautionary actions such as replication [2].

The objective of this paper is to develop a fault tolerance architecture model which has high reliability, availability, robust and also complete. In our proposed architecture solution we aim to hide occurred faults from users and enable them to continue their work with no hindrance in spite of a fault happening. In our solution we aim to integrate proactive measures and reactive measures into our architecture, thus, increasing its reliability and availability significantly. In our architecture design, we used a separate replication module to check the reliability of virtual machines from time to time and we introduced a fault detection module which performs multiple actions to maintain the availability of virtual machines.

2. Literature Survey

Many Fault Tolerant Architectures were reviewed for this report. The literature related to the Architectures was reviewed for a better understanding of the existing solutions. The architectures are summarized below:

2.1. MAP-Reduce

MapReduce[3] is a programming model and an associated implementation for processing and generating large data sets, which is a major part in Hadoop. Few years ago, Google needed to process a huge amount of unstructured data within reasonable time, so a distributed system was required. There were some issues of parallelizing the computation of data that was distributed and fault tolerance was needed. MapReduce was designed to solve these issues. Map and Reduce are basic functions of MapReduce model. Data is a set of input and output key/value pairs. In Map stage, the Maps accept input and process intermediate key/value pairs. In Reduce stage, they shuffle and reduce functions, which takes an intermediate key pairs and merges the key value to make new key/value. After processing, the data is stored in HDFS. Job Tracker, Task Tracker and Job history server control MapReduce procedure. Job tracker controls jobs to be done. Task tracker is helper for job tacker. Job history log made by job controller is stored on job history server.

There are workers in Map and Reduce phase. The master assigns map and reduce to workers and controls them. Every map and reduce task has states which are idle, in-progress and completed. After Reduce phase, the outputs are made into files which are then combined. Pings are sent to workers by the master. If there is no response from a worker during a certain amount of time, the worker is checked by the master. Other workers which completed their tasks are set to idle task for rescheduled. Completed tasks can re-run for failures. In case of master fail, the master makes periodic checkpoints, so if the master die, the last checkpoint is used for recovery. If there is a only single master, fault can not be recovered. This architecture is suitable for huge amounts of data runs on thousands of machine, and requires a lot of clusters.

2.2. HAProxy

HAProxy[4] is an open source application which is used for web load balancing and acts as proxy server for applications based on TCP/HTTP. HAProxy distributes load in several web servers and support fault tolerance. Github, Stackoverflow, Reddit and many number of web sites use HAProxy. The key features of HAProxy are load balancing, proxying and monitoring. HAProxy periodically performs health checks of server by requesting health check information from servers .If one server dies, HAProxy pulls the server out of available server list. Then, HAProxy redirects the request to anotheravailable server. This system uses job migrations and replication policies.

This method is only useful for application which use TCP/HTTP session where several server resources are required, and is not ISSA level fault-tolerance architecture.

2.3. BFT- Byzantine Fault Tolerant Cloud

Cloud infrastructure are mostly divided into two types. First one is well-provided and well-managed infrastructure by large cloud companies such Amazon, Microsoft,Google, etc. Another is user-contributed computing resources called voluntary-resource. These type of cloud resources are very dynamic and low priced, but there are less reliable and less powerful. Also connections among user-contributed nodes are not trusted. Nodes provided by large cloud companies are connected by high speed cables, but there are unpredictable connection links in user voluntary-resource cloud environment. The user-contributed environment is small and weak for fault-tolerance, so Byzantine Fault Tolerant Cloud(BFT-Cloud)[5] architecture was invented to address the limitations. Replication strategy is used in BFT cloud, and 3f replicas are used to handle faults on nodes. There is one primary and 3f replicas ,so this can guarantee f nodes faults.

Cloud applications consist of several modules which are deployed on cloud nodes. Each module performs a certain job for cloud application.

Voluntary-resources consist of user computing resources which are heterogeneous and less reliable. To handle this limitation, the replication is used for fault tolerance. If a request requirement is sent by the cloud module, one node is selected as primary and other nodes are selected replicas. One primary and other replicas run the request in the BFT group. All response is collected from all nodes in the group after a certain amount of time. If responses are different among nodes, fault tolerance procedure is executed and performs primary or replica update. This system requires user resources, so high network bandwidth is also required and user computer can be on low performance. The fault detection ability is 33% because of 3f +1 which guarantees maximum f faulty. If there are 1000 nodes, 333 abnormal nodes can be detected in this system.

2.4. GOSSIP

Gossip architecture[6] is an improvement of the Byzantine architecture. It uses replication to identify a fault in the cloud environment. Every node has a decision vector, and selects a neighbor node and uses decision vector to update node's state. If there is a faulty node which produces undesirable outputs, all other nodes are notified that there is a conflict. The fault detection ability is 50%. For example if there is 2f+1 node in the network, this architecture can detect f node faults.

2.5. MPI

Every node in cloud computing architecture have to run the long process which may result in some errors, that cause the loss of data, if the faults are not effective managed. Message Passing Interface (MPI)[7] is a standard framework for parallel programming which use the checkpoint/restart and job migration techniques to handle the faults. Positive message will be sent from the faulty node to the healthy node then the faulty node migrates the program running from the last checkpoint to the healthy node to continue the process. The architecture of MPI divided to two layers. While the top layer is independent of the infrastructure communication, the bottom layer which is called SSI (Systems Services Interfaces) responsible for specifying whether the backup is needed or not.

2.6. LLF - Low Latency Fault Tolerance

The low latency fault tolerance (LLFT)[8] provides fault tolerance capability for distributed application in a local-area network. The main components of this architecture can be listed as Low latency message protocol, Leader-determined membership protocol, Virtual determinize framework. By using the Leader/Follower replica strategy provides the low-latency Fault Tolerance with minimum of the response time that ignores the user recognition. To achieve this, LLFT uses the Low latency messaging protocol which means the message will be transfer from the primary replica to the backup replicas frequently within a group. The replicas can interact with each other via a group multicast, with this approach the communication between groups will be ensured. The leader-determined membership provides the capability of having a consistent view of membership within a group by selecting the leader of the group deterministically based on the precedence and ranks in a group.

2.7. Vega Warden Architecture

Vega Warden[9] attempts manage the security and resource overhead in the Cloud System due to multiple User Management due to multiple instances of VMs, Cloud Applications running on shared resources.

- Usability – The cloud tenants and end users have different identities that are maintained in the system. It influences the performance of the whole system and User Experience.
- Security – Same resources and different instances of the cloud is a security threat because of the same default privileges on the system.

Vega Warden tackles this problem by a centralized User Management Architecture -

It decentralizes the User Authentication system and makes creates instances of it on VM environment. A user is like a Sub-letter of Cloud Servers and User Service is common and it manages users globally. The local Resource Controller works on each node and provides a uniform R Interface, and the providers authenticate the users and apply their own policies.

Decentralization is achieved through Naming Service. It divides the cloud into different virtual cloud instances. There are mainly two kinds of authentication services that run here:

- User Service that's global
- Local auth services for VIP and ASP
- There are many VM instances for VIP apps auth is direct.
- Whereas for ASP there are RControllers who wrap up the resources and act as a gateway for authorization and access control. NSS for VIP and RC for ASP work on client side.

2.8. AFTRC Architecture

The AFTRC Architecture [10] basically focuses on the time taken by systems to respond because in real time environments, if response is after certain time, it has no value.

It has the following modules that are connected to each other: AT Acceptance Test: It checks if the output of a certain application is valid or not. If it is valid it passes it to TC – Time checker – It just calculates the time take by the system to respond. RC – Reliability checker will calculate the reliability of the VM based on AT and TC. It is calculated with an algorithm. DM Decision maker will make a decision based on RC.

RC Recovery cash stores the checkpoints in case you have to revert back to last known state.

The main focus is, to include more reliability factors on which decisions are to be made and it will be more effective. Also one resource manager is working i.e. proactive resource manager we can also use reactive resource manager which will not remove the node but try to resolve the problem which causes node failure

2.9. FTWS

FTWS[11] is a proposed architecture which comprises a fault tolerant workflow scheduling algorithm for providing fault tolerance by using replication and resubmission of tasks based on the priority of the tasks in a heuristic manner.

The architecture of FTWS in a cloud environment has storage servers and computational servers in cloud environments. Storage servers are mainly related to data storage and do not require mapping of services but on the other hand computational servers offer services associated to computing resources which involve mapping of services to tasks. The major modules present in an FTWS process are Pre-processor module(PM), Replication based Scheduler Module(RSM), Executor Module(ResEm) with rescheduling if necessary and a Data Scheduler(DS).

The Workflow as follows: First the user submits their workflow with a cut off, replication factor and a resubmission factor in the form of abstract data structure format to the pre-processing module. The PM now produces the DAG based on the data and control dependencies between them and separates the tasks based on computational or storage services. The PM also generates a threshold which is used for prioritizing the tasks. The RSM replicates the tasks based on the level of priority. After mapping,

ResEM sends the tasks to the servers and it starts a timer based on the expected execution time. If the ResEm receives an output, it moves on to dependent tasks or else it resubmits the task to the servers.

2.10. FT-CLOUD

In the FT-CLOUD architecture[12] for FT-Cloud, the system designer provides the early design of the cloud application and a component graph is made. Then a component ranking algorithm is used to calculate the significant values. The components can now be ranked using these significant values. The most important components in the cloud application can be recognized. For each of the components identified the most suitable fault tolerance strategy is selected. Then the enhanced design of the cloud application and the component ranking outcomes are delivered back to the system designer.

2.11. MAGI-CUBE

The architecture of magi-cube[13], the system is built on top of HDFS, which is used as a storage system which works for reading and writing files and also metadata management. Other than this a file splitting and repair component has been developed which works in the background self-reliantly from the HDFS for fault tolerance. Even though these two components are two separate parts in this system, they communicate with each other during splits distribution and accessing of k splits for file repairing.

2.12. FTM

FTM[14] is one of the reactive techniques. In this architecture, the Fault Tolerance Manager system which has two main components. The first one is FTM Kernel, this component is responsible for making decisions in which method of fault tolerance will be used. The next one is Messaging monitor module which contains 4 sub-components namely Replication manager, Fault detection/Prediction Manager, Fault Masking Manager and Recovery Manager. This component exchanges and monitors the messages among the replica as well as any modules of this structure. The first sub-component is Replication Manager which is responsible for replicating the application instance in desired nodes. The next sub-component is Fault Detection/Prediction Manager which is responsible for predicting and managing the faults before they happen. The FTM Kernel based on the results from the fault detection stage will select the appropriate recovery method by sending a notification to the Fault Masking Manager component and the Recovery Manager. In Fault Masking Manager, the masking procedures will be applied to faults to prevent it from resulting into errors, which can harm the availability of the service. Recovery Manager sub-component acts as a complementary support for Fault detection/ prediction manager and Fault Masking manager, the purpose for this is to increase the system's lifetime by applying the recovery mechanism to resume the faulty nodes. Besides, there are two components which are Client interface, the one being responsible for communication between the user and FTM system. The other one is Resources manager, which is responsible for monitoring the working state of the physical and virtual resources, providing the significant balanced resources costs and performance for Cloud service provider.

2.13. SUMMARY

Every Fault Tolerance system has one drawback or the other. For instance, Map-reduce and Magi-Cube are specialized for only big data processing. HA proxy can provide fault tolerance, but focuses more on load balancing. It is not a platform level fault tolerance but application level fault tolerance. BFD-cloud system is less powerful but more reliable. Similarly other solutions are also specialized for specific purposes, and they can only handle either proactive method or a reactive method at a time. That is, other architectures can handle only one way to recover from a fault, after the fault has happened or avoid the fault from happening. A much more robust architecture for fault tolerance in cloud computing is required and also that provides high reliability and availability. No solution has the HA feature.

3. Proposed Solution

To improve reliability and availability, the proactive and reactive methods are combined into a new model for a more complete Architecture. The fault handling mechanisms are divided into two parts. In the first mechanism, the heartbeat protocol is used to check if a replica set composed of VM clusters is alive or not. If a fault is detected, fault recovery mechanisms are executed which are composed of checkpoint/recovery, job migration and restart. The second way for fault handling is to predict a fault. If a job which is executed on VM clusters is finished, the result which is pass or a fail is notified, and the reliability factor is calculated based on the result. If a job passes, the reliability factor increases, and if a job fails, the reliability factor decreases. When the reliability factor reaches the already specified minimum reliability, the replica set is not trusty and replaced with another replica set.

The significance of two feeds(i.e., heartbeat and VM output) to the Replication manager is that in case of a Network Failure, the Reliability Calculator will give incorrect values. Here, the heartbeat ping will help in determining the Reliability of the Node.

Many architectures were reviewed for this research to help in implementation of the new architecture, and amongst them, AFTRC and FTM architectures are majorly referred.

4. Development

Advance Cloud Fault Tolerance(ACFT) provides two mechanisms of fault handling i.e., Reactive and Proactive. Both the mechanisms run in parallel on different threads but are connected to each other.

Reactive FT comprises of fault detection and three levels of recovery mechanisms. ACFT watches each cloud nodes(VMs) by heartbeat protocols invoked through a fault detector which checks their liveliness. If a fault is detected, the fault recovery system tries to fix the fault by executing the FT mechanisms in the following order: 1.Checkpoint/recover 2.Job migration 3.Restart.

The Proactive FT runs in parallel to the reactive FT.

Proactive depends upon the reliability of the node to invoke the Fault Tolerance protocol. The VM Group and the Fault Detector continuously feed the Replication Manager with Success and Failures of services running on the VM, which in turn affects the Reliability of the node. If the Reliability of the node goes below minimum Reliability, the node is replaced.

4.1. The Architecture

ACFT has six main components:

1. Cloud Provider Interface ,
2. Fault Decision Maker
3. Fault Detector
4. Fault Handling
5. VM Replica Group
6. Replication Module

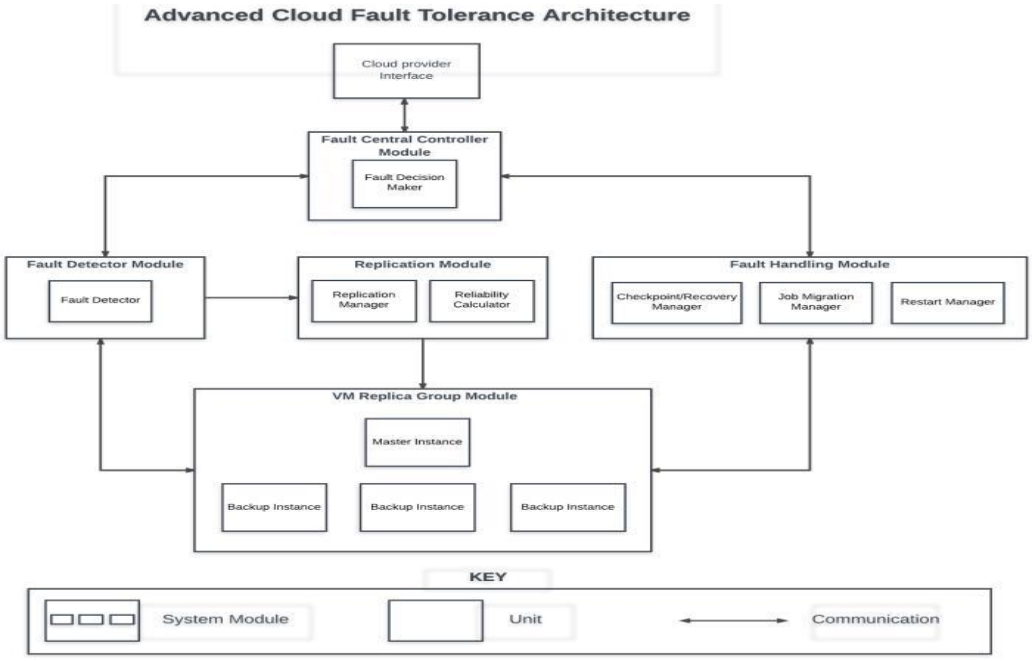

Fig. 2. An overview architecture of ACFT

4.1.1. Cloud provider Interface: The cloud provider interface provides the clients with an automatic requirement configuration. An automated configuration can reduce human error and the client can be notified of the error to select recovery protocols if a fault happens. The User can also set configuration for recovery policy via this interface. The CCI provides the configuration file with parameters namely minReliability, maxReliability and Reliability Factor.

4.1.2. Fault Decision Maker: This module is composed of the fault tolerance solution and controls the fault handling components such as replication manager, checkpoint/recovery manager, job migration manager and restart manager. If a fault occurs, the fault detector notifies the fault detector which executes the fault handling unit. The fault decision maker then cascades the fault to one of the three modules present. For example, If a fault occurs, the fault decision maker launches a recovery module in par with the recovery policy from the configuration. If user does not set any configuration, checkpoint/recovery manager is invoked first. If the fault cannot be handled, the fault decision maker executes job migration and restart modules respectively. If the fault still cannot be recovered, all clusters are marked as failed in a replica set or reliability of the replica set goes under the standard limit which is also logged in the configuration. The decision maker requests replication manager to delete/replace the replica set on the list.

4.1.3. Fault Detector: Replica set is composed of master and backup nodes. The fault detector continuously communicates with master node in the replica set by using heartbeat protocol. The fault detector waits for its response and if there is no response after a specified time, it assumes that the master node is dead, and the module reports it to the fault decision maker.

4.1.4. Fault Handling Modules: If the fault is notified to the fault decision maker, the module launches checkpoint/recovery manager first. The master node which failed tries to revert to the checkpoint which are periodically saved in the master node. If the fault cannot be recovered, the job migration module is invoked. The job which ran on master node is moved onto one of the backup nodes, and the backup node becomes master node.If that also fails to fix the fault, the final solution is

the restart module , which loses the job that ran on the cloud and the job starts from the beginning. If all solutions fail, the replica set is removed from the list by the Replication manager with other Replica and the result is logged.

4.1.5. VM Replica group: It comprises of one master and three backup nodes in one replica set (Fig. 3). The master node exchanges heartbeat message with the fault detector. The Replica Group is the major module of ACFT.

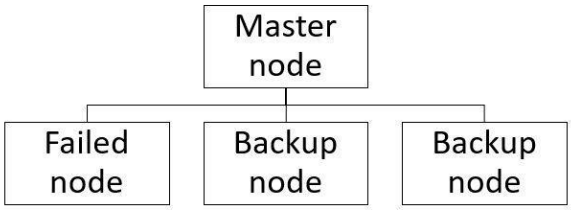

Fig. 3. VM Replica group

4.1.6. Replication manager: The replication manager controls replica sets. This module has the list of replica sets and if there is a fault which cannot be recovered, replication manager removes the replica set from the list. Users can make a configuration in which the number of replication sets via cloud provider interface is mentioned.

4.2. Evaluation

ACFT can be evaluated by reliability and availability aspects of Fault Tolerance which is divided into two factors. the first factor is job migration. When a job migration is executed, the master node is changed. As one replica set is composed of one master and three backup, the replica set guarantees that job migrations can only be done thrice, which means that only 33 percent is covered. To improve reliability, checkpoint/restart scheme is added before job migration by default configuration option.
In case of failure of job migration, restart scheme, which is a feature of High Availability system is also added. Another aspect is to check minReliability. The basic concept is that if faults occurs frequently and the faults are fixed, the replica set cannot be trustworthy hence the Proactive approach is used here. There are two replica sets assume that one can pass and another can fail. The reliability_factor is 0.03. Because of weight, the reliability of fault is decreased rapidly. It means if the replica continues to fail, the reliability decreases faster . Fig 4 and 5 shows the Algorithms for reactive and proactive methodologies in ACFT.

4.2.1. Availability: ACFT provides high availability. The recovery mechanism guarantees job without loss. Checkpoint/recovery manager makes VM recover the previous checkpoint which is marked periodically. One replica set is composed of one master and three backup nodes, so if the master is not recovered via checkpoint/recovery, backup nodes can take the job instead of master via job migration module. Because of three backup nodes, it will guaranteesupto three fault fixing which checkpoint/recovery did not fix. Through two solutions, high availability are ensured without the job loss.

*4.2.2. Reliability:*If the reliability of a replica falls below minReliability, the replica set is removed on the list in the replication manager. There are three factors in configuration: *reliability, reliability_factor, minReliability ,maxReliability.*

If a replica set execute a job without fault, the reliabilityincreases, but if a fault happens, the reliability decreases.Initial value of reliability is 1. For example, if reliability_factor is 0.2 and minReliability is 04, reliability falls to 0.8 after the fault. The replica set is removed if the reliability falls under 0.4.

4.3. Fault Tolerance Algorithm

The ACFT Algorithms is divided into two threads that run in parallel to each other. Fig. 4 shows the thread 1 that is specific to Reactive FT methodology. There is an infinite loop which checks if any VM is dead. As soon as it discovers a dead Service, the fault recovery function is executed which tries to recover the fault through different Fault Fixing Methodologies until it is deemed unrecoverable and logged.

Fig. 5 describes the algorithm which runs on thread 2 in parallel to thread 1. The Algorithm has a while infinite loopand a variable called reliability. The reliability calculator is fed with the Fail / Pass values through the Hearbeat_ping and vm_result. If the Reliability falls below the minReliability at any time after a consecutive number of cycles. The node is replaced.

The Reliability calculator relies on the min and max Reliability and Reliability for calculations of Reliability.

```
while (forever) {
    hearbeat_ping = get_response(hearbeat_protocol);
    // get heartbeat_ping
    if (heartbeat_ping >= dead_time)
        // if VM does not respond for more than dead_time
        fault_recovery_protocol();
    // execute fault recovery protocol
}

function fault_recovery_protocol() {
    return = fault_detection_manager(checkpoint);
    if (return = fault){
        return = fault_detection_manager(job_migrate);
        if (return = fault){
            return = fault_detection_manager(restart);
            if (return = fault){
                log(node.unrecoverable);
                break;
            }
        }
    }
}
```

Fig. 4. Runs on thread 1

```
input: minR, maxR, reliability_factor
// get these values from the configuration File
while (forever) {
    reliability = reliability_cal(vm_result, hearbeat_ping, reliability_f)
    // Calling calculator function for each cycle.
    if (realiability < minR) {
        // if reliability falls below minR after n cycles
        node_replace_protocol();
        // execute protocol to replace the node
    }
}

function reliability_cal(vm_result, hearbeat_ping, reliability_f) {
    // Function definition for calculating Reliability
    reliability = depends_on(vm_result, hearbeat_ping);
    if (not fault)
        reliability = realiability + ( reliability * reliability_f);
    else
        reliability = realiability - ( reliability * reliability_f);

    if (reliability >= maxR) // reliability exceeds Max Value
        reliability = maxR
}

function node_replace_protocol() {
    replica_manager.(replica); // Replace the node
}
```

Fig. 5. Runs on thread 2

5. Viewsets

As an essential part of this paper, the following architectural design for the ACFT system are illustrated as below:

5.1. Logical View

The ACFT system sequence diagram is shown in Fig. 6, which describes the reactive and proactive sequences. First, the Fault Detector unit keeps sending the Heartbeat message to every single instance in each replica node in VM Group modules. VM Group responds to the Heartbeat message to the Fault Detector unit, after a pre-set time, if the responding time from the VM instance in VM Group is greater than the pre-set time, the instance will be consider at a fault which means the fault happened. The Fault Detector sends failed message to the Fault Decision maker and the Replication module. When the Fault Decision maker receive the message, it invokes the Fault recovery process. The first fault recovery mechanism is triggered by Fault Decision Maker unit then sends the fix request to Checkpoint/Recovery Manager unit. Afterwards, the Checkpoint/Recovery Manager executes the process to recover the instance to the latest checkpoint. Then a message will be send back to Checkpoint/Recovery Manager to ensure the fault is fixed or not. The fault status will be sent to Fault Decision Maker. If the fault is still existing, the next job will be sent to Job Migration unit and Restart Manager unit in order to re-enable the availability to the system using the similar procedure.

The other flow of the system is the proactive process. When there is no response to a heartbeat request, the Fault detector also sends a fault message to Replication module. The reliability decreases in the module if a fail message is received. If the reliability goes below minimum reliability, the replacement of the replica set is executed. When the replication module receives a pass message which means there is no fault in the job, the reliability increases.

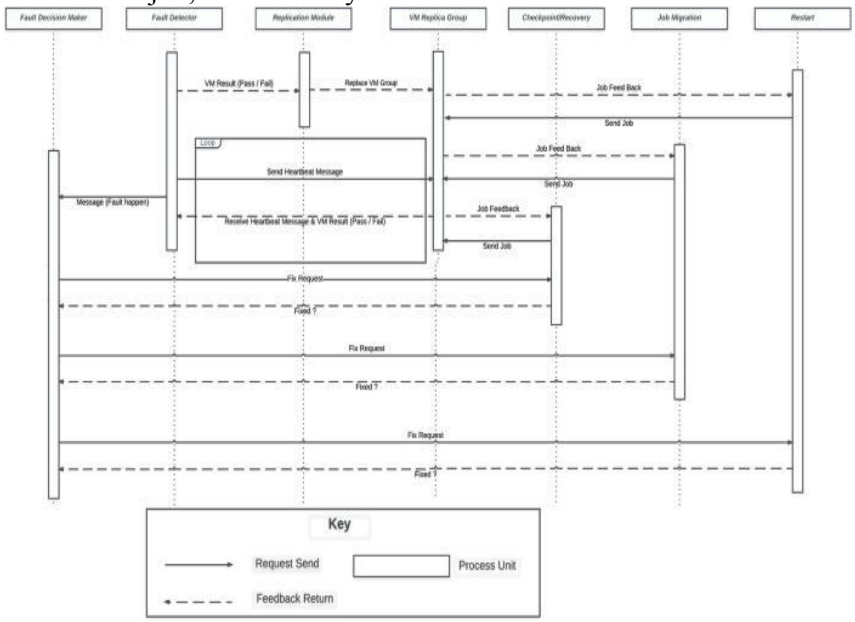

Fig. 6. ACFT sequence

5.2. Process View

The two process views describe the dynamic behavior of the reactive and proactive fault tolerance in proposed ACFT system.

*5.2.1. Reactive:*The main flow of the process begin with the Heartbeat message send to VM Replica Group module in order to detect the fault. The purpose of this process is to checking the status of every instance in VM Replica Group. If the fault is detected by Fault Detector unit, the process will continue following the red flow. The fault will be handover to Faut Decision Maker unit to execute the Fault recovery mechanisms as described as light blue color flow. The Fault Decision Maker is

programmed to sequential run the Fault recovery mechanism as follow : Checkpoint/Recovery, Job Migration and Restart to VM Replica Group to overcome the fault.

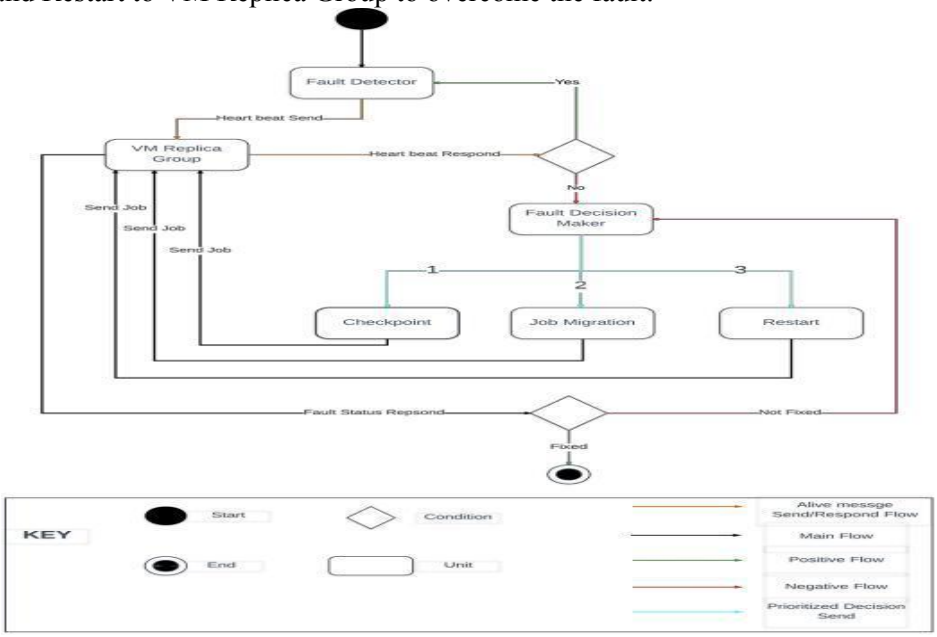

Fig. 7. ACFT reactive

5.2.2. Proactive: Fig. 8 illustrated the processing flow of Proactive method embedded into ACFT. First, VM Replica group send response Heartbeat messages and Pass messages,. When Fault Detector unit receives the messages, the messages are sent to the Replication Manager. The Replication manager invokes the Replication Calculator unit which makes the Reliability increase or decrease depending on a job pass or fail. The replication manager replace the VM Replica group with another when the Reliability become less than the minimum reliability.

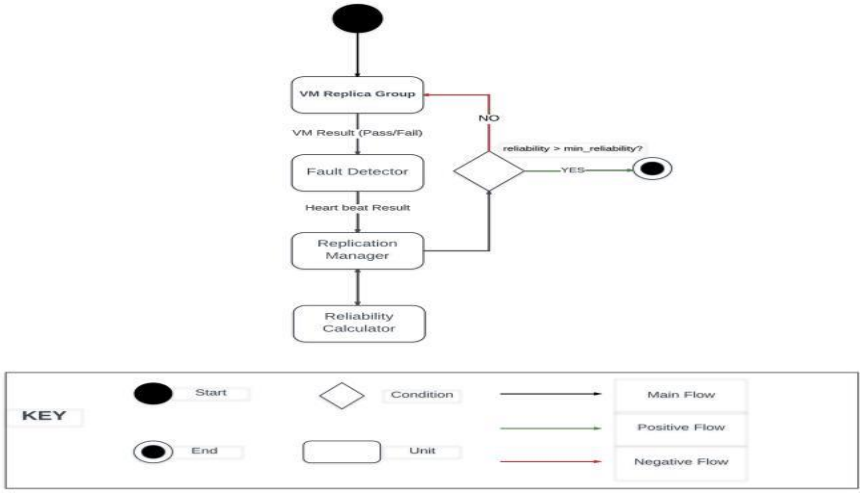

Fig. 8. ACFT proactive

5.3. Development View

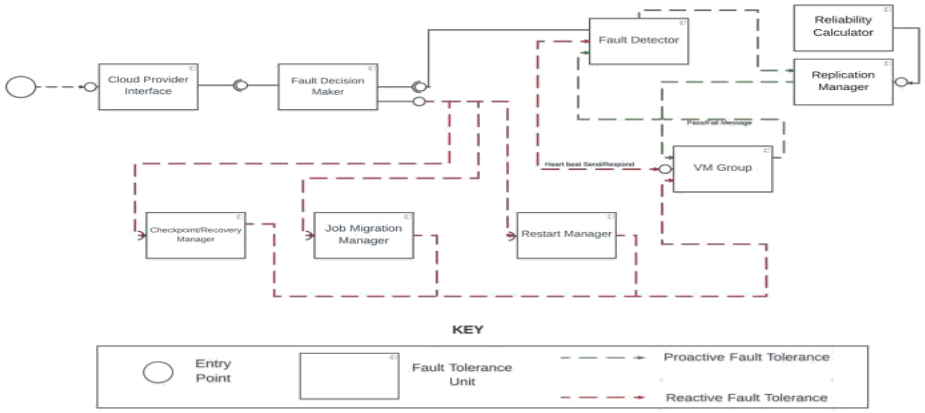

Fig. 9. ACFT components

The development view shown above depicts our system from a developer's perspective. This view also known as the implementation view focuses on the association of the actual software modules in the development environment. In the Fig. above we can see how each module works with its related modules and if that module is depending on another module as well.

5.4. Physical View

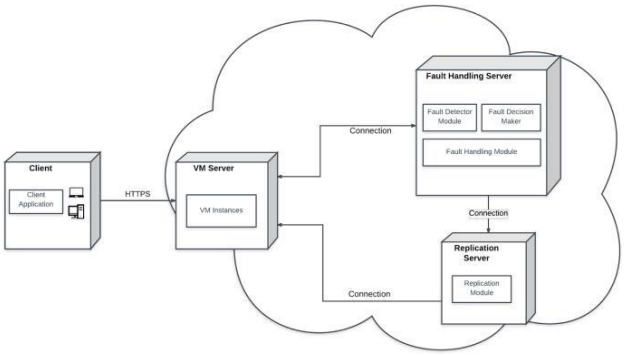

Fig. 10.ACFT deployment

The deployment diagram in Fig. 10 defines the application architecture at a physical level. This view depicts the system from a system engineer's perspective. It shows the topology of our proposed architecture and shows all the components on the physical layer. The components in our physical view are client, VM server, fault handling server and the replication server. The fault handling server and the replication server have other modules included in them.

5.5. Scenarios

5.5.1. Reactive:

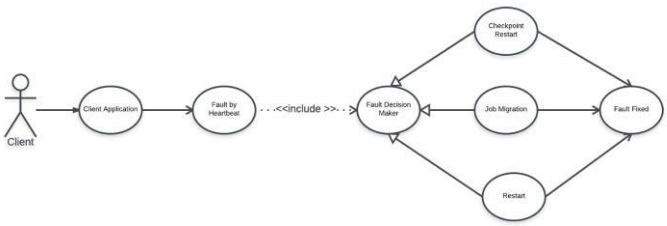

Fig. 11. Reactive Use Case Diagram of ACFT

Fig. 11 shows the Reactive Use Case Diagram of ACFT. The Client Interacts with the Client Application and feeds theheartbeat at regular intervals, in case of a fault the fault detector assigns it to the checkpoint/recovery manager which tries to fix the fault, if it doesn't it cascades it to the next module.

5.5.2. Proactive

Fig. 12. Proactive User Case Diagram of ACFT.

Fig. 12 shows the Proactive User Case Diagram of ACFT. The Client Interacts with the Client Application and feeds the data to the fault detector and the reliability is calculated and a decision is made whether to replicate the VM or not.

6. Future Issues & Conclusion

Further to this proposed architecture solution, there are few areas of development that can be looked in to in the future. In our architecture, in the replication module the only quality attribute that is increased is reliability but in the future we would like to enhance the functionality of this module. In our current architecture, when the fault decision maker detects a fault it first invokes the checkpoint/recovery module, if that does not fix the fault then it invokes the job migration manager and if that does not fix the fault then it finally invokes the restart manager. This works in sequence as in one module after the other, but in the future we would like our fault decision maker to decide the most appropriate method to fix the fault and directly invoke the respective module. The fault decision maker should decide the module to invoke based on the type of fault and the simplest way to fix it.

In our fault handling module, we have the job migration manager, a limitation in the job migration manager is that it for a particular VM group the job can be migrated only thrice. This is because a VM group has one master and three backups so job migration can only happen thrice, in the future we would like to increase the number of times a job can be migrated efficiently.

We would also like to develop and create a simulation of our architecture and evaluate it against other existing fault tolerant architectures to study the differences and how we can further improve our proposed architecture.

To conclude we looked into various existing fault tolerant architectures and introduced an innovative architecture which is a combination of proactive and reactive fault tolerant architectures. This architecture is a great option to be used as a fault tolerance mechanism for cloud computing. It has all the advantages that clients usually require, it has dynamic behaviour of reliability, its ratio of availability is very high and it is highly fault tolerant.

7. References

[1] Mell P and Grance T 2011 The NIST definition of cloud computing.

[2] Egwutuoha I P, Chen S, Levy D and Selic B 2012 A fault tolerance framework for high performance computing in cloud. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* pp. 709-710. IEEE Computer Society.

[3] Dean J and Ghemawat S 2008 MapReduce: simplified data processing on large clusters. *Communications of the ACM***,** vol **51** no 1 pp.107-113.

[4] Kaushal V and Bala A 2011 Autonomic fault tolerance using haproxy in cloud environment. *Int. J. of Advanced Engineering Sciences and Technologies*, vol **7** no 2 pp.54-59.

[5] Zhang Y, Zheng Z and Lyu M R 2011 BFTCloud: A byzantine fault tolerance framework for voluntary-resource cloud computing. In *Cloud Computing, 2011 IEEE International Conference on* pp. 444-451. IEEE.

[6] Cheraghlou M N, Khadem-Zadeh A and Haghparast M 2016 A survey of fault tolerance architecture in cloud computing. *Journal of Network and Computer Applications **61*** pp.81-92.

[7] Kaur J and Kinger S 2013 Analysis of different techniques used for fault tolerance IJCSIT: *Int J Comput Technol* vol **5** no 3 pp. 737–741
[8] Zhao W, Melliar P M and Mose L E 2010 Fault tolerance middleware for cloud computing. *In: Proceedings of the 2010 IEEE 3rd international conference on cloud computing* pp. 67–74.
[9] Xiaoyi Lu J, Yu L, Zou Y, and Zha L 2010 Vega Warden: a uniform user management system for cloud applications. *In: Proceedings of the 2010 fifth IEEE international conference on networking, architecture, and storage* pp. 457–464.
[10] Malik S and Huet F 2011 Adaptive fault tolerance in real time cloud computing. In *Services, 2011 IEEE World Congress on* pp. 280-287 IEEE.
[11] Jayadivya S K, Nirmala J S and Bhanu M S S 2012 Fault tolerant workflow scheduling based on replication and resubmission of tasks in Cloud Computing. *International Journal on Computer Science and Engineering* vol **4** no 6 p. 996.
[12] Zheng Z, Zhou T C, Lyu M R and King I 2010 FTCloud: A component ranking framework for fault-tolerant cloud applications. In *Software Reliability Engineering, 2010 IEEE 21st International Symposium on* pp. 398-407. IEEE.
[13] Feng Q, Han J, Gao Y and Meng D 2012 Magicube: High Reliability and Low Redundancy Storage Architecture for Cloud Computing. In *Networking, Architecture and Storage, 2012 IEEE 7th International Conference on* pp. 89-93. IEEE.
[14] Jhawar R, Piuri V and Santambrogio M 2013 Fault tolerance management in cloud computing: A system-level perspective. *IEEE Systems Journal* vol **7** no 2 pp. 288-297.