FAULT TOLERANCE APPORACH FOR CRASH FAILURES IN CLOUD ENVIRONMENT

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Rajeev Kapoor
(Roll No.821132003)

Under the supervision of:
Dr. Anju Bala
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2015

CERTIFICATE

I here certify that work which is being presented in the thesis entitled ," Fault Tolerance approach for crash failures in Cloud environment ", in partial fulfillment of the requirements for award of Master in Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Anju Bala and refers other researcher's work which are duly listed in the reference section.

The Matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature

Rajeev Kapoor

This is to certify that above statement made by the candidate is correct and true to the best of my knowledge.

Dr. Anju Bala

Assistant Professor

CSE Department

Thapar University

Patiala

Countersigned by

10/06/2015

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Patiala

(Dr. S.S Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

CERTIFICATE

I here certify that work which is being presented in the thesis entitled, "Fault Tolerance approach for crash failures in Cloud environment", in partial fulfillment of the requirements for the award of Master of Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Anju Bala and refers other researcher's work which is duly listed in the reference section.

Matter presented in the thesis has not been submitted for the award of any other degree of this or any other University.

Signature

Rajeev Kapoor

This is to certify that above a statement made by the candidate is correct and true to the best of my knowledge.

Dr. Anju Bala

Assistant Professor

CSE Department

Thapar University

Patiala

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Patiala

(Dr. S.S Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

Acknowledgment

I express my sincere and deep gratitude to my guide Dr. Anju Bala Assistant Professor of Computer Science and Engineering Department, Thapar University Patiala, for invaluable guidance, support and encouragement. She provided me all resource and guidance throughout thesis work.

I would also like to thank all staff members who were always there at the need of an hour and provided with all the help and facilities, which I required for the completion of my thesis. At last but not the least I would like to thank God and mine parents for not letting me down at the time of crisis showing me the silver lining in the dark Clouds.

Rajeev Kapoor
821132003

Abstract

Fault tolerance is an important issue in the Cloud Computing which is mainly concerned with providing the guarantee of availability and reliability of a system. It also checks a system that must work its functions in a proper manner and data must not lose even some crash / some failure of components of the system occurs. Therefore, Fault tolerance of Cloud Computing provides an environment in which a system can work in a corrective manner using the fault handle techniques. The main benefit of implementing fault tolerance of Cloud Computing is to provide the failure recovery of the components, the cheapest cost of handling fault and also metrics for improving performance of a system. In Cloud Computing, resources are available as services like Software as service (SAAS), Platform as resources (PAAS) and Infrastructure as a service (IAAS) on private and public networks. Cloud Computing has the ability to use applications on the Internet that store and protect data while providing a service. It can be used as storage Cloud to hold application, business, and personal data.

In this thesis, various fault tolerance techniques have been explored to handle the crash failure of Name Node, further, the analysis of cost effective tools like HA proxy server and NFS of Linux. NFS is the repository for Name Node System data in the Cluster. The framework for the fault tolerance approach has been purposed that will handle crash failures of Name Node in the Cluster.

In this framework, VM Workstation has created a cluster of Linux nodes. In this cluster, a Node HAProxy server has been installed to monitors all servers which have been providing the services on Hadoop. Various Hadoop Name servers have been created for handling the crash of Name node. These nodes have acted as Backup Name Nodes Server for fault tolerance. There has been one Linux server which has provided the services of NFS share for common repository Hadoop server File System metadata.

In the thesis, use of data scheduling, job migration and data replication technique have given the solution. HAProxy has been used for handling migration of jobs of Hadoop servers in the case of Name Node Failure. HAProxy has been migrated jobs from one Name Node to another Name Node in the case of Name Node Failure. NFS Server has been provided the common repository for Name Node File System. Using this proposed Framework, services of Hadoop Server has been resumed with any loss of data. The experimental results show that the use of load balancing and distributed files system techniques handles the framework of the Name Node crash for this purpose.

List of Tables

List of Figures

Table of Contents

Chapter 1
Introduction

In this chapter, the basic introduction about Cloud Computing, characteristic of Cloud Computing, Cloud Computing Stack and deployment model details are described in detail. The end of this chapter gives the structure of the thesis.

1.1. Cloud Computing

Cloud computing is a term used for Internet based service by Industry Stack Holder (like Google, Amazon.com in late 2006. It provides on demand service with quick implementation, low cost, less staff and the quick solution of business needs. Cloud Computing is an evolution of Grid Computing. The Grid Computing is developed for scientific purposes, now Cloud Computing is used for commercial purposes. Cloud Computing is based on a "pay per use" pricing model. On the other hand, Grid Computing is a collection of heterogeneous recourses in the distributed system. It is such a distributed system in non-interactive mode. Firstly, one must join the Grid, and then he is allowed to access and contribute power to other members in the grid. Cloud Computing is a large pool of resources such as hardware, development platforms and services which are available in virtual resources form. Now Cloud Computing becomes Computing power into a public utility which is an effective tool for the Industry and society as a whole.

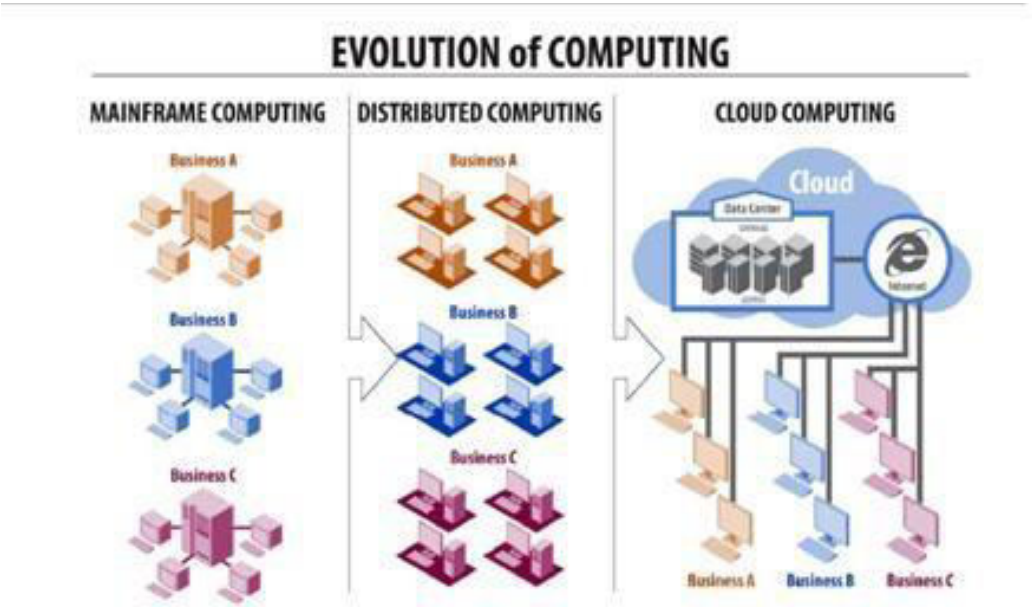

Figure1: Evolution of Cloud Computing [1]

In Figure 1 shows, Cloud Computing has been developed after two phases 1. Mainframe Computing 2. Parallel Computing.

The National Institute of Standards and Technology (NIST) describes in this way, " Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [1]

In the NIST Cloud Computing model, the computing resources like network, servers, storage, applications and services are available on the requirement basis from Cloud Computing service providers. The user can rapidly get and release resources with minimal management effort or service provider interaction.

According to NIST Cloud model, it includes five essential characteristics**,** three service models, and four deployment models which are given in detail below.

1.2 Characteristics of Cloud Computing

As per NIST definition, every Cloud must include five essential characteristics which are given as below.

- **On-demand self-service:**
  A Customer can increase / decrease computing capabilities automatically without human interaction. It means that the customer can increase/ decrease server time, storage and network capacity automatically by just giving a request. The service provider system provides the services on demand immediately on customer request. [1]

- **Broad Network Access:**
  These services are also available on a huge range of network medium and access though using standard mechanisms. This standard mechanism promotes usage of heterogeneous thin clients or other clients like workstations, mobile phones and tablets. [1]

- **Resource pooling:**
  The service provider creates a pool of resources with the help of this model. In this model, different physical and virtual resources are dynamically assigned and reassigned as per customer demand. This gives the notion of location independence in which customer is not able to control and does not have knowledge of the exact location of provided resources and services. They have only knowledge up to higher level abstraction like country, state and data center etc. [1]

- **Rapid Elasticity:**
  A Customer has become capable of increase or decrease of resources as per demand. The customers appear this facility as unlimited and appropriated in any quantity at any time. [1]

- **Measured service:**
  Cloud system provides provisions to automatically control and optimizes resources on Cloud network. It leverages resources with metering capability. It means that resources can be monitored, controlled and reported. It also provides transparency for the service provider and customer of using services. [1]

1.3. Cloud Computing Stack

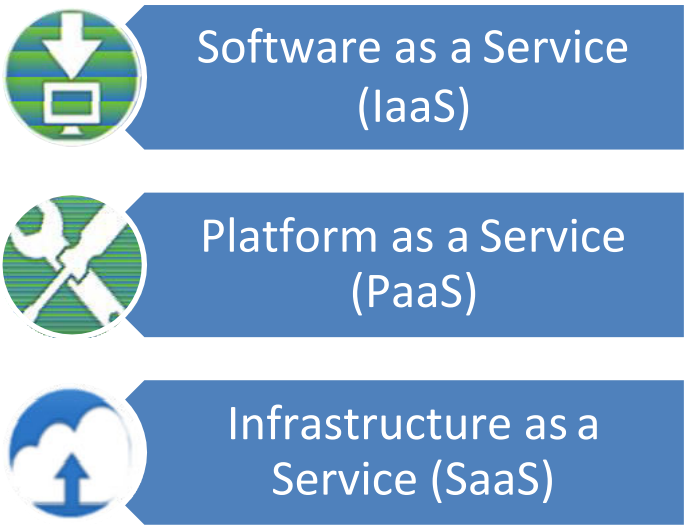

Figure 2: The service model of Cloud Computing [1]

1.3.1 Software as Service (SaaS):

Software as Service is the top abstraction layer of Cloud Computing. In this type of service, the end user applications are delivered as service. In this environment, multiple clients on Cloud use its services. The best example of it is Goggle Appe offering basic business service mail and word processing. Here, Platform and infrastructure is abstracted. It can be deployed and managed with minimum efforts. Consumers can purchase the ability to access and use an application or service that is hosted in the Cloud. [1]

1.3.2 Platform as Service (PaaS):

Platform as Service (PaaS) is the middle level abstraction layer of Cloud Computing. It is a software level abstraction in which application platform (like Java, .Net, Web servers) is deployed at some server and this platform services are used at the remote site on Cloud network. Applications can be built and deployed more expensively. Consumers only purchase access to use the platform, that is enabling them to deploy own software and applications in the Cloud. It saves money and time of the customer to develop the platform to run its application. Multiple clients are available on the cloud. The customer itself does not manage the operating system access the network. The platform provides as service. It includes databases and development tools, all delivered as a service via the Internet. PaaS offers some specific platform which includes programming language or APIs, such as Java or Python. [1]

1.3.3 Infrastructure as Service (IaaS):

Infrastructure as service (IaaS) is a low level abstraction layer service of Cloud Computing. It provides basic storage, computing capabilities and networking as services over the network. Servers, storage systems, switches, routers, and other devices are pooled together and delivered as a service. From this pool, required components are made available to the customer end in handling the workload. It also provides high-performance computing facility to the customers which save the expenses of the customer to deploy the dedicated system. Infrastructure as the service uses the concept of virtualization. Virtualized servers that provide a highly available on-demand infrastructure. Therefore virtualization, management and operating system software are part of IaaS. [1]

1.4. Deployment Models

There are four basic deployment models private Cloud, community Cloud, public Cloud and hybrid Cloud which are described in detail:

- **Private Cloud:**
  Private Clouds are exclusive build for a single organization. It provides almost all control over data, security and quality of service to the client end. The end client has own infrastructure and has control on the applications are deployed on it. It can be built and managed by own IT Technicians or by Cloud providers. This gives a high level of control over the use of Cloud resources. Data store in the private Cloud can only share with others on the on trust basis. [1]
- **Public Cloud:**
  Public Clouds are built to provide general public. It is based on "Pay-as-You-Go" Model and third parties run this cloud services. Public Cloud is mixed services like Cloud servers, storage and network services. They provide a way to reduce customer risk and cost. It provides a flexible and temporary extension to enterprise infrastructure. [1]

- **Hybrid Cloud:**
  Hybrid Cloud uses combined features of both private and public Cloud Computing. It offers more flexibility, control and security of multiple deployment models. Hybrid Cloud provides the flexibility to deploy resources on premises and remote site. The organization only pays for extra when compute resources are used at the remote site. [1]

- **Community Cloud:**
  Community Clouds are built for the organizations with similar interests and requirements. Third party manages and hosts them internally. The cost of the establishment of this Cloud is shared among all organization with similar interest. It reduces the Cost and easy implementation of solution. For example, all Government agencies with the same interest share Cloud services, but other private organizations cannot use these services. [1]

1.5 Issues and Challenges in Cloud Computing

- **Security and Privacy**
  The Security and privacy of data is the major issue in Cloud Computing. The major challenge is how to secure data from loss and prevent from attacks, unwanted changes from hacker / an intruder. When data movement occurs from one station to a remote site on the network, there is a possibility to loss of control of data, network data poisoning and other threats may occur. Cloud Computing constitutes the policy for privacy concern because it prevents accidently or debatably change or remove of the data from internal and external enemies.

- **Performance**
  The second major issue in Cloud Computing is the performance. The Cloud must provide improved performance when a user specially uses Infrastructure as service. In the running cloud, performance is normally measured with the checking of the capability number of applications. The major challenge is to improve performance. It can be achieved by rescheduling policy, improved processing algorithms and monitor services for providing good performance.

- **Reliability and availability**
  The strength of any system is based on the degree of reliability and availability of resources which provides. So the strength of Cloud is also based on the degree of reliability and availability of Cloud. Reliability means that resources are available without disruption. The degree of reliability will go down when any resource gets down. To achieve this level, the redundant resource must be provided. Availability is denoted as the plausibility of obtaining resources, whenever they are needed for usage. To provide high reliability and availability, Cloud must efficiently handle the problems like denial of service attacks, performance slowdowns, equipment disorder and natural disasters.

- **Scalability and Elasticity**
  Scalability and elasticity is the most unique feature of Cloud Computing. Scalability means the ability of system performing well even when the resources have been scaled up. The elasticity means the dynamic integration and extraction of physical resources to the

Infrastructure. In Cloud, scalability will be provided by adding a new computer to an existing service provider to a single node in the system.

- **Interoperability and Portability**
  Interoperability is the ability to use the same applications across various Cloud service provides platforms. It provides flexibility of migrating in and out and switching to Cloud whenever they want without vendor lock in the period. Cloud portability denotes that one solution can work on one system and can run on another platform also.
- **Resource Management and Scheduling**
  Resource Management deals at various levels, namely hardware, software, virtualization, security and other parameters. It includes the management of memory, disk space, threads, VM images and other parameters. Job scheduling in Cloud Computing is portioning of jobs in parallel tasks, interconnection of Clouds and processors, assigning priority to jobs and selection of processors or Cloud to allocate jobs, memory allocation and task execution monitoring.
- **Energy Consumption**
  Cloud data center is the house of thousands of servers and also an arrangement of the cooling infrastructure. For cooling these servers, they use a huge amount of energy and produces greenhouse gases (GHGs). Here, the goal is not only to reduce the consuming energy, but also to maintain environment standards.

1.6 Organization of Thesis

Chapter 2 - This chapter describes the literature Survey in the detail.

Chapter 3 - This chapter describes the problem statement of the thesis.

It gives the gap analysis, problem statement and mythologies are discussed.

Chapter 4- This chapter describes the design and implementation of the proposed Framework and description of tools used for implementation.

Chapter 5- This chapter describes an experiment result of implementation of proposed framework. It includes snapshots of the experimental results.

Chapter 6 - This chapter describes the conclusion, contributions of work done and future Research work possible.

Chapter 2
Literature Review

Previous chapter provides a brief introduction about Cloud Computing and its important issues. This chapter discusses about the previous research work done in the area of fault tolerance techniques in Cloud and also covers the research gaps and requirements to improve fault tolerance on different parameters.

2.1 Fault Tolerance

Fault tolerance is a major topic in the Cloud computing which is mainly concerned with providing the guarantee of availability and reliability of a system. It also checks a system that must work its functions proper a manner and data must not lose even some crash or some failure of components of the system. Therefore, Fault tolerance of Cloud provides an environment in which a system can work corrective manner with their fault handling techniques. The main benefit of implementing the fault tolerance of Cloud Computing, it provides the failure recovery of the components, cheapest cost of handling fault and also provides metrics for improving performance of a system.

2.2 Metrics for Fault Tolerance in Cloud Computing

There are many types of fault exists in Cloud Computing, there are various metrics to measure fault in Cloud Computing. Some important metrics are discussed below in the table.

Table 1: Metrics for Fault Tolerance in Cloud Computing

Synod	Metrics for Fault Tolerance in Cloud Computing	Description
1.	**Proactive fault tolerance**	It uses the avoidance policy, in case of any fault ,error and failure occurs then it replaces suspected component the new one.[10]
2.	**Reactive fault tolerance**	It uses the reaction policy on any fault, error and failure occurs then it takes an immediate action for correction of fault by using one of methods like check point, Restart, Replay and Retry etc.[10].
3.	**Adaptive**	It acts according to the situation, all the procedures are done automatically.[10]
4.	**Performance**	It is used to measure the efficiency of the System. [10]
5.	**Response Time**	**It** is the amount of time taken to respond to a

		particular algorithm.[10]
6.	**Scalability**	Scalability means the ability of the system perform well even when the resources have been scaled up.[10]
7	**Throughput**	This is used to calculate the number of tasks whose execution has been completed.[10]
8	**Reliability**	This aspect aims to give the correct or acceptable result within time bounded environment.[10]
9	**Availability**	The probability that an item will operate satisfactorily at a given point with in time used under stated conditions. The availability of a system is typically measured as a factor of its reliability as reliability increases, so does availability.[10]
10	**Usability**	The extent to which a product can be used by a user to achieve goals with effectiveness, efficiency, and satisfaction.[10]
11	**Overhead Associated**:	It determines the amount of overhead involved while implementing a fault tolerance algorithm. It is composed of overhead due to the movement of tasks, inter-processor and inter-process communication. This should be minimized so that a fault tolerance technique can work efficiently.[10]
12	**Cost effectiveness**:	Here the cost is only defined as a monitorial cost.[10]

Based on the literature survey, there are various metrics for fault tolerance in Cloud Computing. These various metrics (as shown in the Table 1) are used to measure fault in Cloud Computing. On the basis of these metrics, the metrics will be considered for a particular problem statement, find the techniques to solve the problem statement and prepare the framework for implementation is fault tolerance for crash handling of the name node server of Hadoop. The various fault tolerance techniques are discussed in the Table 2.

2.3 Fault Tolerance Techniques for Cloud Computing

Table 2: Fault Tolerance techniques

Synod	**Techniques for Fault Tolerance in Cloud Computing**	**Description**
1	**Check pointing**	It is an efficient task level fault tolerance technique for long running and big applications .In this scenario after doing every change in

		System a check pointing is done. When a task fails, rather than from the beginning it is allowed to be restarted that job from the recently checked pointed state.[10]
2	**Job Migration**	Some time it happened that due to some reason a job can- not is completely executed on a particular machine. At the time of failure of any task, the task may migrate to another machine. Using HA-Proxy job migration can be implemented.[10]
3	**Replication**	Replication means copy. Various tasks are replicated and they run on different resources, for successful execution and for getting the desired result. Using tools like HA-Proxy, Hadoop and AmazonEc2 replication can be implemented.[10]
4	**Self- Healing**	A big task is divided into parts .These will be multiplications are done for better performance. When various instances of an application are running on various virtual machines, it automatically handles failure of application instances.[10]
5	**Safety-bag checks**	In this case the blocking of commands is done which have not met the safety properties.[10]
6	**S-Guard**	It is less turbulent to normal stream processing. S-Guard is based on rollback recovery. S-Guard can be implemented in HADOOP, Amazon EC2.[10]
7.	**Retry**	In this case we implement a task again and gain. It is the simplest technique that retries the failed task on the same resource.[10]
8	**Task Resubmission**	A job may fail now whenever a failed task is detected, In this case at runtime the task is resubmitted either to the same or to a different resource for execution.[10]
9	**Timing check**	This is done with the help of watchdog. This is a supervision technique with time of critical function .[10]
10	**Rescue workflow**	This technique allows the workflow to persist until it becomes unimaginable to move forward without catering the failed task.[10]
11	**Software Rejuvenation**	It is a technique that designs the system for periodic reboots. It restarts the system with clean state and helps to fresh star.[10]

12	**Preemptive Migration**	Preemptive Migration count on a feedback-loop control mechanism. The application is constantly monitored and analyzed.[10]
13	**Masking**	After employment of error recovery, the new state needs to be identified as a transformed state. Now if this process is applied systematically even in the absence of effective error, it provides the user error masking[10]
14	**Reconfiguration**	In this procedure we eliminate the faulty component from the system.[10]
15	**Resource Co-allocation**:	This is the process of allocating resources for further execution of the task.[10]
16	**User specific (defined) exception handling**	In this case user defines particular treatment for a task on its failure[10]

.

When the problem statement is declared, the next task will be to find effective techniques, algorithms and tools to find an effective solution of the problem statement. In Table 2, various techniques for fault tolerance in cloud Computing are discussed. On the basis of nature of the problem, suitable technique is chosen for the solution of the problem statement. In this thesis, job migration and replication technique is used for solving the problem statement. The HA Proxy tool is used for job migration and NFS file system uses the replication of the master Name node File System. NFS replication of Name node data is used for handing the loss of data due to the crash of Name Node Servers in the proposed framework.

2.4 Comparison among various models based on protection against the type of fault, and procedure

Table 3 Comparison among various models of Cloud Computing [10]

Model Name	**Protection against the type of fault**	**Applied procedure for tolerate the fault**
AFTRC	Reliability	Delete node depending on their reliability, Backward recovery with the help of check pointing
LLFT	Crash-cost, the trimming fault	Replication
FTWS	Dead line of work flow	Replication and resubmission of jobs
FTM	Reliability, availability, on demand service	Replication of user application, in case of replica failure, use algorithm like gossip based protocol.

CANDY	Availability	It assembles the model components generated from IBD and STM according to allocation notation. Identify the relationship between action in activity SNR and state transition in system SRN synchronizes activity SNR to system SRN.
VEGA-WARDEN	Usability, security, scaling	A two layered authentication and a standard technical solution for the application
FT-CLOUD	Reliability, crash and value fault	The Significant component is determined on the basis of ranking. Optimal FT technique is determined.
MAGI-CUBE	Performance, reliability, low storage cost	Source file is encoded in then splits in to save as a cluster. The File recovery procedure is triggered and the original file is lost.

When techniques and algorithms are chosen for the implementation of the proposed framework. It is necessary to compare these techniques with other techniques for comparison purposes, In the Table 3; there is a description of comparison among various models based on protection against the type of fault, and procedure.

2.5 Challenges for fault tolerance in Cloud Computing

Providing the fault tolerance for Cloud Computing is the very serious, complex and challenging job. There are many issues which needs fault tolerance. It needs in depth analysis of complexity, inter- dependability of the problem. The section describes details about various obstacles for the g r o w t h of cloud computing along with measure to overcome these obstacles.

- **Business continuity and availability of services**
  Organizations are always worried about the utility and availability computing services. This provides the opportunity for multiple vendors to provide Cloud Computing service.
- **Data safety**
  Organizations are always worried about whether the utility computing service will be providing data security and data safety. This challenge gives the responsibility to opt adopt different backup strategies for data safe. This provides the opportunity for users to opt other computing methods than Cloud Computing.

- **Performance Predictability**
  The performance is the core challenge in Cloud Computing. There are many virtual machines are share CPU and Memory of the Physical server. The heavy load on the physical machine will reduce the performance of all virtual machines. This needs improvement of the policies to handle a limited number of virtual machines on the physical server.
- **The scalable storage**
  Next challenge is to provide scalability, data durability and high ability to the customer who uses the utility computing services on Cloud Computing. Fault tolerance is needed for its solution.
- **Bugs in the large distributing system**
  The next challenge is to minimize the bugs in the large distributing system. It is a big challenging job to provide bugs free services to the customers. There is need to use tools for handling bugs in the large distributing system.

2.6 Tools Used For Implementing Fault Tolerance

Fault tolerance challenges and techniques have been implemented using various tools. HA Proxy is used for server failure in the Cloud. SHelp is a lightweight runtime system that can survive software failures in the framework of virtual machines. It may also work in Cloud environment for implementing check pointing. ASSURE introduces rescue points for handling programmer anticipated failures. Hadoop is used for data intensive applications, but can also be used to implement fault tolerance. Techniques in Cloud environment like Amazon Elastic Compute Cloud (EC2) provide a Virtual Computing environment to run Linux-based applications for fault tolerance.

One of the main goals of Hadoop and HDFS is to be highly fault tolerant. Therefore HDFS can be spread over hundreds or thousands of nodes. These nodes contain cheap, low- cost hardware. While considering thousands of computer components and hundreds of network devices such as switches, routers, and power units that are involved in these large distributed systems, it causes failures to be very frequent. In these systems, failures can occur daily. So we need a robust fault tolerance essential for a distributed system such as Hadoop. Hadoop and HDFS center its fault tolerance on data redundancy, which is to replicate data so that if one replica is lost then there are backup copies.

Hadoop is an open-source software framework implemented using Java and is designed to be used on large distributed systems. Hadoop is a project of the Apache Software Foundation and is a very popular software tool due, in part, to it being open-source. Yahoo has contributed to about 80% of the main core of Hadoop, but many other large technology organizations have used or are currently using Hadoop, such as, Face book, Twitter, LinkedIn, and others. The Hadoop framework is comprised of many different projects, but two of the main ones are the Hadoop Distributed File System (HDFS) and Map Reduce. HDFS is designed to work with the Map Reduce paradigm. This survey paper is focused around HDFS and how it was implemented to be very fault tolerant because fault tolerance is an essential part of the modern day distributed systems.

Chapter 3
Problem Statement

After studying various research statements in literature review, the following problems can be defined. Further, this chapter focuses on research gaps, a problem statement and methodologies to solve them.

3.1 Gap Analysis

There is much advancement in the field of Cloud Computing achieved by researchers. But there are some gaps still existing Cloud Computing. The topic of the thesis is related to find a gap for implementing the distributed databases like Hadoop. The Hadoop is self-fault tolerance distributed file system on Cloud Computing. Besides it is self-tolerance, still it needs an improvement. The major weak point of Hadoop is the fault tolerance.

It handles a crash of the Master Name Node in the cluster perfectly. There is need to promote the Secondary Server to Master Name Node which delays services. All services are broken during this period. It still needs to improve this system handling such type of crash in Hadoop.

3.2 Problem Statement

The underlying problem is the handling of the crash of Master Node by using third party tools and services. When the master node fails, it takes times to promote the secondary master node to the main master node. But it is not simple and a quick solution to provide services without interruption.

Hadoop and its file system provides fault tolerance on data redundancy, it creates multiple replica on different nodes. In the case, one node is lost, other backup copies are available. Hadoop provides Secondary name server for Load balancing and replica of the main Name Node server. However, there is no provision to handle the situation.

In our problem definition, Fault tolerance is based on multiple name node replicas, so HAPROXY server will handle the fault tolerance. When the Hadoop Name server is going down, this server automatically transfers all requests to another server and also provides a common replica of Name nodes.

3.3 Methodologies

- To handle fault at the Name Node Server end, HAProxy has been used. It redirects the request to another server if one server fails.
- To provide the same name node data to the NFS share folder. It provides a common repository for all nodes.
- All Name Nodes replicas have the same point to access data.

Chapter 4

Design and Implementation

This chapter discusses the solution to the problem which has been defined in the previous chapter. Further, methodologies have been described to implement the solution.

4.1 Proposed Design of Solution

This section presents the proposed design of the problem discussed in the previous chapter. The proposed design is described with the help of the figure given below

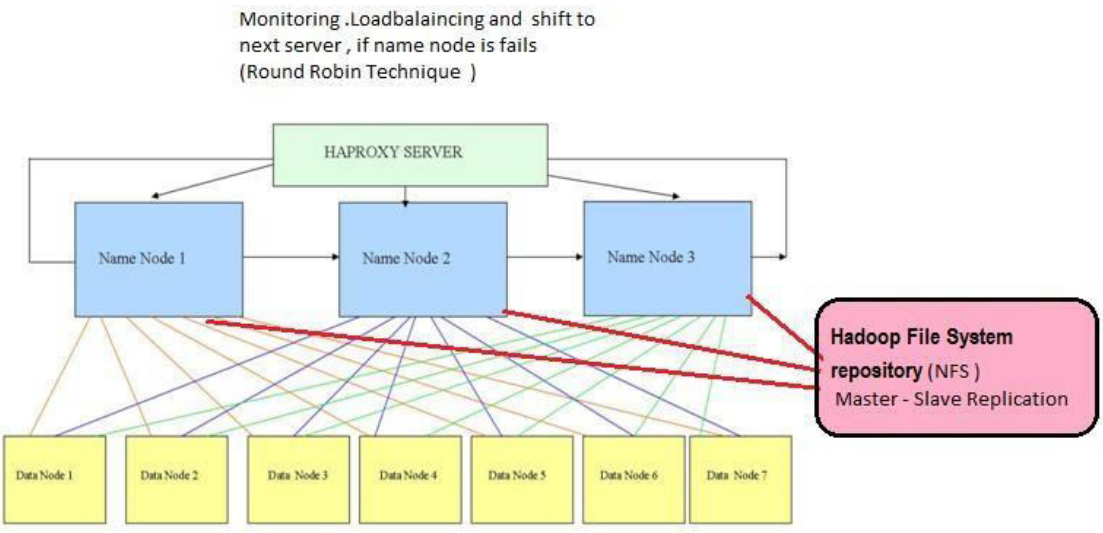

Figure 3: The proposed framework for fault tolerance of Hadoop Nodes

Above said Figure 3 shows one HAPROXY Server, three Hadoop Name Node Server and seven data nodes in test design. HAPROXY monitors the entire three Hadoop name nodes. All of these have a replica; if one Name Node is failed then HAPROXY reschedules the next Name Node. All three Name nodes have the same Data Nodes. This solution provides effective fault tolerance against Crash Failures of the Name Nodes.

The further design is prepared for common the repository of Name Node files. It provides high availability of the same file system data. In the case of fault occurs, HA Proxy migrates from one node to another node; it provides the same replica of the previous name node.

For creating the Common replica of Name Node Server, Network File System NFS is used. Design of the common repository is given below

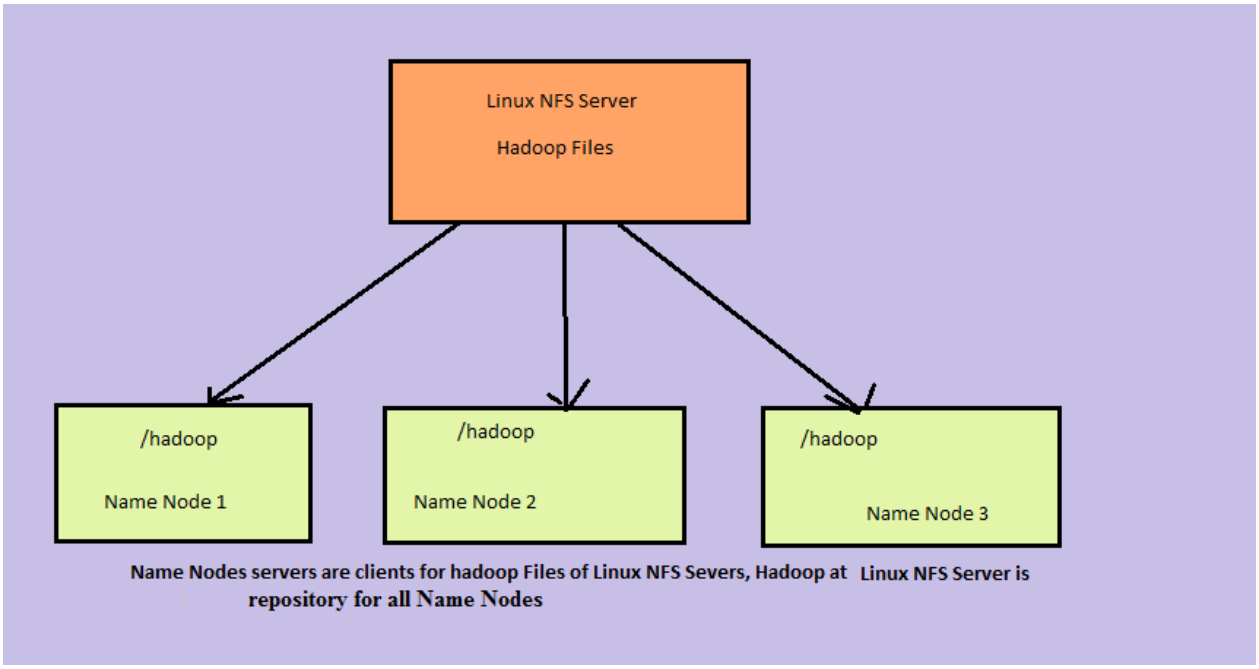

Figure 4: The Proposed framework for the backup repository of Hadoop File System at NFS Server.

In above figure, there is one NFS server, which shares the directory which is accessible of one NFS clients. There are three Name Node sites which are using NFS share directory as the Hadoop main storage directory. The main directory of Hadoop of each name node is mounted to the NFS share directory on Linux NFS Server.

This proposed design gives a solution of effectively handling of the crash for Master Nodes in the Hadoop System. The services are resumed without any interruption or need to promote any secondary server for resuming their services.

4.2 Framework Design

The following section gives the overall guidance on the implementation of fault tolerance framework using various tools in Cloud environment. In the above proposed design in Figure 1, there is need to develop such framework system for implementations. There is various software and tools are required to develop this Framework. In this Framework, we are using Hadoop 2.0, Apache HA Proxy, Fedora Linux 20 and Red hat Linux for NFS Server.

4.3. Implementation of the proposed framework

4.3.1 VM Workstation

VMware Workstation is a hypervisor that runs on x86-64 computers; it enables users to set up one or more virtual machines (VMs) on a single physical machine, and use them simultaneously along with the actual machine. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD, and MS-DOS. [2]

Product Benefits:

- VMware Workstation supports bridging existing host network adapters and share physical disk drives and USB devices with a virtual machine. In addition, it can simulate disk drives and mount an existing ISO image files into a virtual optical disc drive so that the virtual machine sees it as a real one. Likewise, virtual hard disk drives are made via vodka files.[2]
- VMware Workstation can save the state of a virtual machine (a "snapshot") at any instant. These snapshots can later be restored, effectively returning the virtual machine to the saved state.[2]
- VMware Workstation includes the ability to designate multiple virtual machines as a team which can then be power on, power off, suspended or resumed as a single object, making it particularly useful for testing client-server environments.[2]
- The VMware Player, a virtualization package of basically similar, but reduced, functionality, is also available, and is free of charge for non-commercial use, or for distribution or other use by written agreement.[2]

4.3.2 HAProxy Version 1.5

HAProxy stands for High Availability Proxy. It uses the tools for load balancing and server failure. Companies do not want their website to go down, or worse, for users to notice the site is down. Due to this, larger websites will run on multiple servers to provide some level of high availability. In HAProxy, there is typically a load balancer to distribute the load among a pool of web servers. It also works as a fault tolerant system. Whenever a server goes down it is taken out of the pool until it is once again ready to handle requests. HAProxy has the ability to perform this task by doing periodic health checks on all the servers in a cluster. Even if one of the application servers is not working, users will still have the availability to the application [3].
HAProxy will properly handle the request from users by redirecting them to the second server, giving the impression that all is well. HAProxy is a free, very fast and reliable solution offering high availability and load balancing mechanism, and proxy for TCP and HTTP based applications. When HAProxy is running in HTTP mode, both the request and the response are fully analyzed and indexed. Thus it becomes possible to build matching criteria on almost anything

found in the contents. However, it is important to understand how HTTP requests and responses are formed, and how HAProxy decomposes them. It will then become easier to write correct rules and to debug existing configurations [3].

HAProxy is installed on all "Front End" Server Templates. A "Front End" (FE) is a server with a static IP (typically an Elastic IP) and is registered with DNS. A normal setup has two "Front Ends." A FE server can also have an application server installed on it or it can stand alone. The FE's purpose is to direct users to available application servers. It can also detect server failures (hardware or software), and provides client transparent fault tolerance [3].

HAProxy can easily migrate to another server in the presence of system failures. HAProxy currently does not support the HTTP keep-alive mode, but knows how to transform close mode. Right now, HAProxy only supports the first mode (HTTP close) if it needs to process the request. This means that for each request, there will be one TCP connection. Logs, content switching and filtering most often cause no problem for persistence with cookie insertion. [3]

HAProxy takes care of all these complex combinations when indexing headers, checking values and counting them, so there is no reason to worry about the way they could be written, but it is important not to accuse an application of being buggy if it does unusual, valid things. It needs very little resource. Its event-driven architecture allows it to easily handle thousands of simultaneous connections on hundreds of instances without risking the system's stability. [3]

The HAProxy Configuration

HAProxy configuration process involves three major sources of parameters:

- The arguments from the command-line, which always take precedence,
- The "global" section, which sets process-wide parameters,
- The sections which can take the form of "defaults", "listen", "frontend" and "backend".

First parameter is an optional list of arguments which may be needed by some algorithms. Here a load balancing algorithm is set to round-robin scheme in the background. The algorithm may only be set once for each background. Secondly, process wide OS- specific parameters are used. They are generally set once for all and do not need to be changed. Our HAProxy configuration file supports keywords for process management and security and performance tuning. Keywords such as stats and _limit- n‘are used for process management and security and for later _Macon‘are used. Then for the third parameter, "defaults" section sets default parameters for all other sections following its declaration. A "frontend" section of HAProxy accepts client connections. A "backend" section of server 1 and server 2 connect to proxy for forwarding incoming connections a "listen" section defines a complete proxy its front and back parts combined in one section. It is generally useful for TCP-only traffic [3]

```
#haproxy.cfg
#--------------------------------------------------------------------------------------------------------------------
  Defaults
  Mode                 http
  Log                 global
  Option               http log
  Option               dontlognull
  Option http-server-close
  Option forward for       except 127.0.0.0/8
  Option               redispatch
  Retries              3
  Timeout http-request    10s
  Timeout queue           1m
  Timeout connect         10s
  Timeout client          1m
  Timeout server          1m
  Timeout http-keep-alive 10s
  Timeout check           10s
  maxconn              3000
  #--------------------------------------------------------------------------------------------------------------------
  Frontend LB
  Bind 192.168.177.100:8080
  reqadd X-Forwarded-Proto:\ http
  Default backend LB
```

```
Backend LB 192.168.177.100:80
Mode http
Stats enable
Stats hide-version
Stats Uri /stats
stats realm Haproxy\ Statistics
stats auth haproxy:redhat          # Credentials for HAProxy Statistic report page.
balance roundrobin                 # Load balancing will work in round-robin   process.
option httpchk
option  httpclose
option forwardfor
Cookie LB inserts
server hadoopserverone 192.168.177.103:50070 cookie hadoopserverone check
server hadoopservertwo 192.168.177.104:50070 cookie hadoopservertwo check
server hadoopserverthree 192.168.177.105:50070 cookie hadoopserverthree check
```

4.3.3 Fedora 20

Fedora 20 is an operating system based on the Linux kernel, developed by the community-supported Fedora Project and sponsored by Red Hat. Fedora contains software distributed under a free and open source license and aims to be on the leading edge of such technologies.[4]

Fedora has a reputation for focusing on innovation, integrating new technologies early on and working closely with upstream Linux communities. Making changes upstream instead of specifically in Fedora ensures that the changes are available to all Linux distributions. The default desktop in Fedora is the GNOME desktop environment and the default interface is the GNOME Shell. Other desktop environments, including KDE, Xfce, LXDE, MATE and Cinnamon, are available and can be installed. Fedora uses the RPM package management system.[4]

Security is also important in Fedora with one specific security feature being Security-Enhanced Linux, which implements a variety of security policies, including mandatory access controls, and which Fedora adopted early on.[4]

4.3.4 Hadoop Version 2

Hadoop which is an Apache project; all components are available via the Apache open source license. Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets [5].

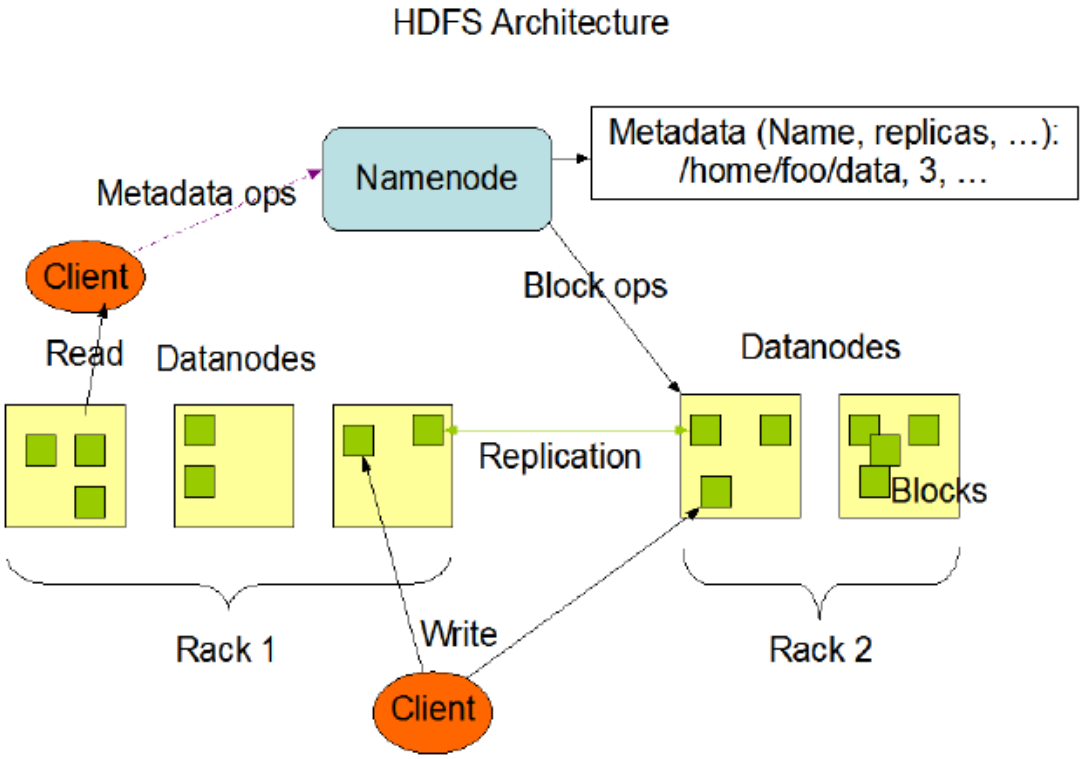

Figure 5 Core Components of Hadoop Architecture [5]

- **HDFS:** A foundational component of the Hadoop ecosystem is the Hadoop Distributed File System (HDFS). HDFS is the mechanism by which a large amount of data can be distributed over a cluster of computers, and data is written once, but read many times for analytics. It provides the foundation for other tools, such as HBase.[5]

- **MapReduce**: The Hadoop main execution framework is MapReduce, a programming model for distributed processing, parallel data processing, breaking jobs into the mapping phases and reduces the phases (thus the name). Developers write MapReduce jobs for Hadoop, using data stored in HDFS for

fast data access. Because of the nature how MapReduce work, Hadoop brings the processing to the data in a parallel fashion, resulting in the fast implementation.

- **HBase:** A column-oriented No SQL database built on top of HDFS, HBase is used for fast read/write access to large amounts of data. HBase uses Zookeeper for its management to ensure that all of its components are up and running.[5]

- **Zookeeper** is Hadoop distributed coordination service. Designed to run over a cluster of machines is a highly available service used for the management of Hadoop operations, and many components of Hadoop depend on it.[5]
- **Oozie A** scalable workflow system, Oozie is integrated into the Hadoop stack, and is used to coordinate execution of multiple Map Reduce jobs. It is capable of managing a significant amount of complexity, basing execution on external events that include timing and presence of required data**.[5]**

- **Pig:** It is an abstraction over the complexity of MapReduce programming. This platform includes an execution environment and a scripting language (Pig Latin) which is used to analyze Hadoop data sets. Its compiler translates Pig Latin into sequences of MapReduce programs.[5]

- **Hive** SQL-like, high-level language used to run queries on data stored in Hadoop, Hive enables developers not familiar with Map Reduce to write data queries that are translated into Map Reduce jobs in Hadoop. Like Pig, Hive was developed as an abstraction layer, but geared more toward database analysts more familiar with SQL than Java programming.[5]

- ***Sqoop:*** Sqoop is a connectivity tool for moving data between relational databases and data, warehouses and Hadoop. The Sqoop balances database to describe the schema for the imported or exported data and MapReduce for parallelization operation and fault tolerance.[5]

- **Flume:** Flume is a distributed, reliable, and highly available service for efficiently collecting, aggregating, and moving large amounts of data from individual machines to HDFS. It is based on a simple and flexible architecture, and provides a streaming of data flows. It leverages a simple extensible data model, allowing you to move data from multiple machines within an enterprise into Hadoop.[5]

4.3.5 The NFS File system

The Network File System (NFS) is probably the most prominent network service using

RPC. It allows you to access files on remote hosts in exactly the same way you would access local files. A mixture of kernel support and user-space daemons on the client side, along with an NFS server on the server side, makes this possible. This file access is completely transparent to the client and works across a variety of server and host architecture [6]

NFS offers a number of useful features:

- Data accessed by all users can be kept on a central host, with clients mounting this directory at boot time. For example, you can keep all user accounts on one host and have all hosts on

  your network mount */home* from that host. If NFS is installed beside NIS, users can log into any system and still work on one set of files. [6]
- Data consuming large amounts of disk space can be kept on a single host. For example, all files and programs relating to Latex and METAFONT can be kept and maintained in one place.[6]
- Administrative data can be kept on a single host. There is no need to use `rap` to install the same stupid file on 20 different machines.[6]

Chapter 5

Experimental Results

In this chapter, two experiments have been performed for getting experiment results. The first experiment is testing Master Name Node failure of Hadoop by using HAProxy Server. The second experiment is the testing a common repository of Name node data by using NFS technology of Linux.

5.1 The experimental result of testing Name Node Failure of Hadoop

In this experiment, there are the total five servers based on the Linux operating system is running on it. The various tools are deployed on this server as per framework requirements. The detailed description is given below:

Server 1: HAProxy Server
This server is used for the monitoring purpose for Hadoop failure. Here HAProxy the monitoring tool is installed on this server. The detail configuration of this server is given below
An IP address: 192.168.177.100 Subnet mask 255.255.255.0
Monitoring Software: HAProxy

Server 2: Name Node 1
This server is used for Name Hadoop. The Hadoop name node date is installed on this server. The detail configuration of this server is given below
An IP address: 192.168.177.103 Subnet mask 255.255.255.0
Monitoring Software: Hadoop Version 2

Server 3: Name Node 2
This server is used for Name Hadoop. The Hadoop name node date is installed on this server. The detail configuration of this server is given below
An IP address: 192.168.177.104 Subnet mask 255.255.255.0
Monitoring Software: Hadoop Version 2

Server 4: Name Node 3
This server is used for Name Hadoop. The Hadoop name node date is installed on this server. The detail configuration of this server is given below
An IP address: 192.168.177.105 Subnet mask 255.255.255.0
Monitoring Software: Hadoop Version 2

Server 5: Data Server

This server is used for providing share folder which is accessible on the entire network. NFS services are installed on this server. The detailed configuration of this server is given below

An IP address: 192.168.177.144 Subnet mask 255.255.255.0

Monitoring Software: NFS Sever

5.2 The Snapshot of experiment results

- **Case 1: Testing whether all servers are working well as per design framework.**
  In this case, we are testing all Hadoop Servers are working well and see the result of HAProxy is giving the correct result.

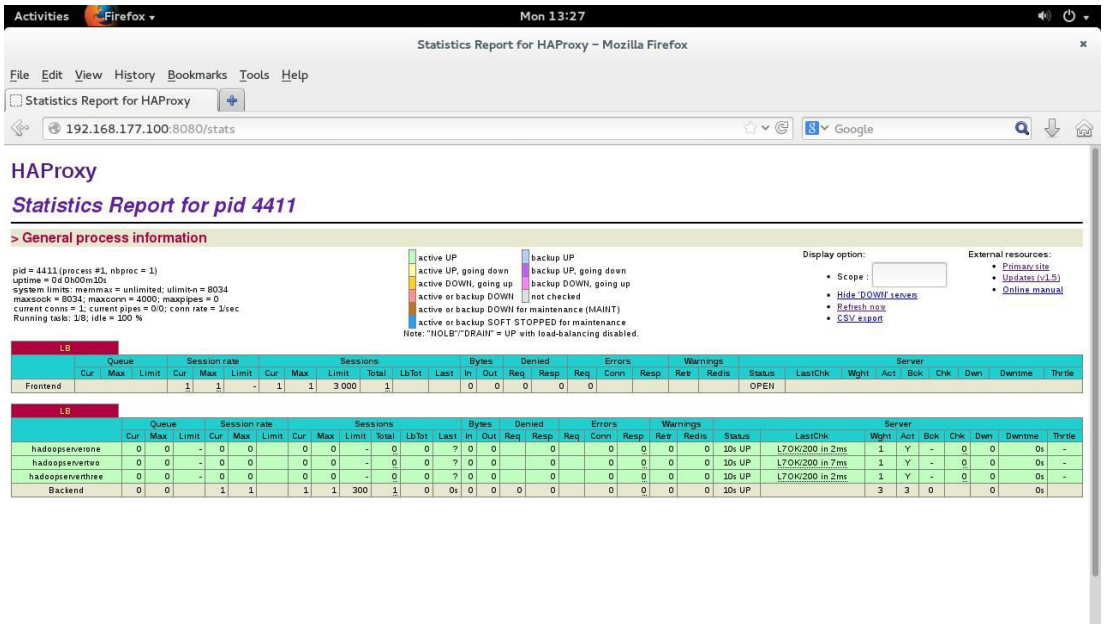

Figure 6: HAProxy status of Hadoop server

The above figure 6 shows, HAPROXY Server showing the all the Hadoop Name Servers are running well. It is ready to monitor all replicas of Hadoop Server on the Framework. It is ready to monitor all replicas of Hadoop Server on the Framework.

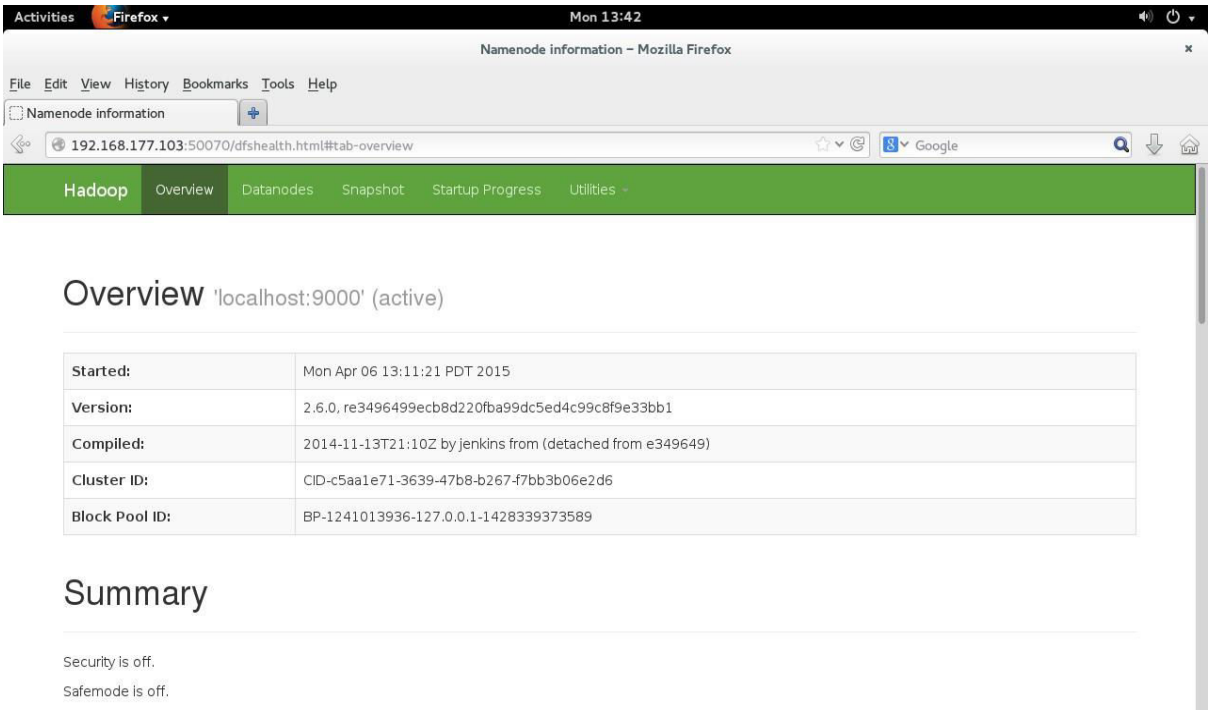

Figure 7: Hadoop Server is running well on Name Node Server1.

The above figure 7 shows that Hadoop Name server is running well. All the services are running well in the state.

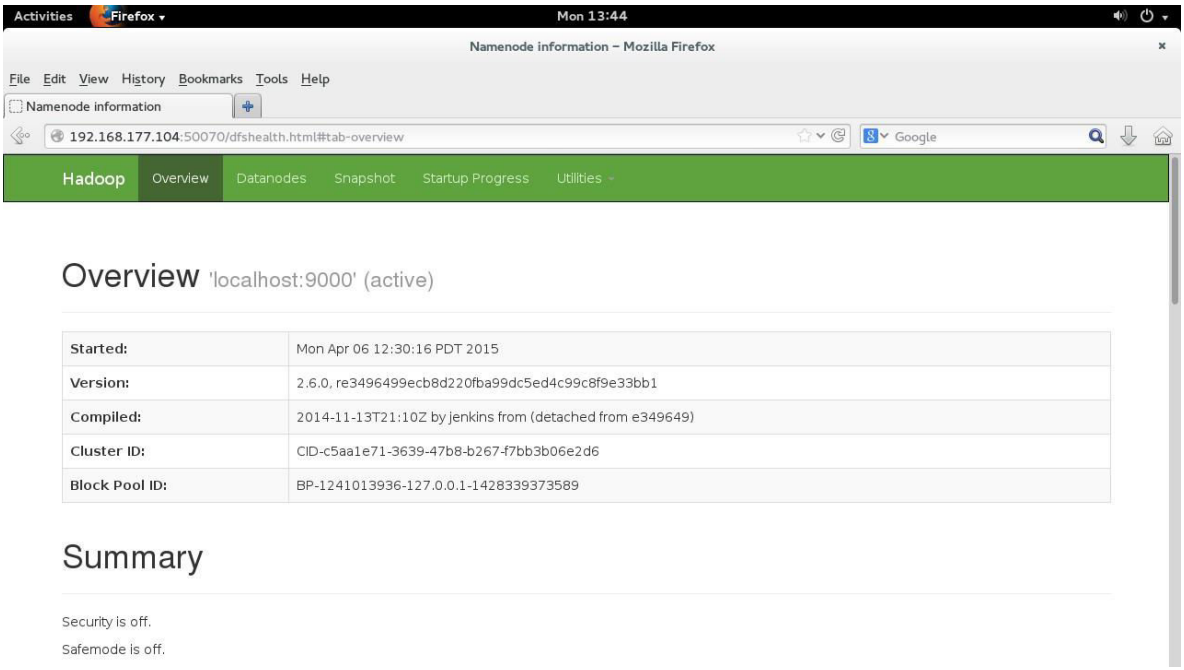

Figure 8: Hadoop Server is running well on Name Node Server 2.

The above figure 8 show the Hadoop Name server is running well. All the services are running well in the state

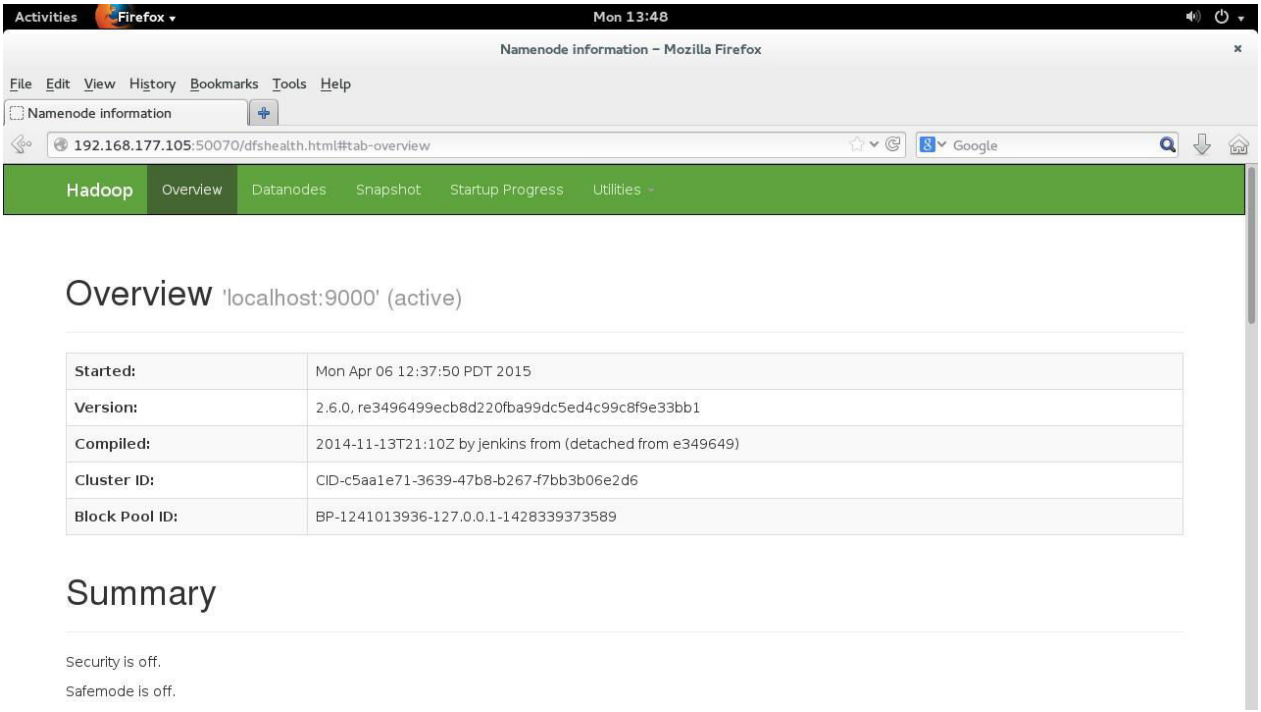

Figure 9: Hadoop Server is running well on Name Node Server 3.

The above figure 9 shows the Hadoop Name server is running well. All the services are running well in the state.

- **Case 2:** Stop the Services of Hadoop Server two and three, HAProxy Server Shows Status as down server.
  In the case, stopping services of Hadoop Name server two and three perform the experiment. It shows two servers are down in the figure.

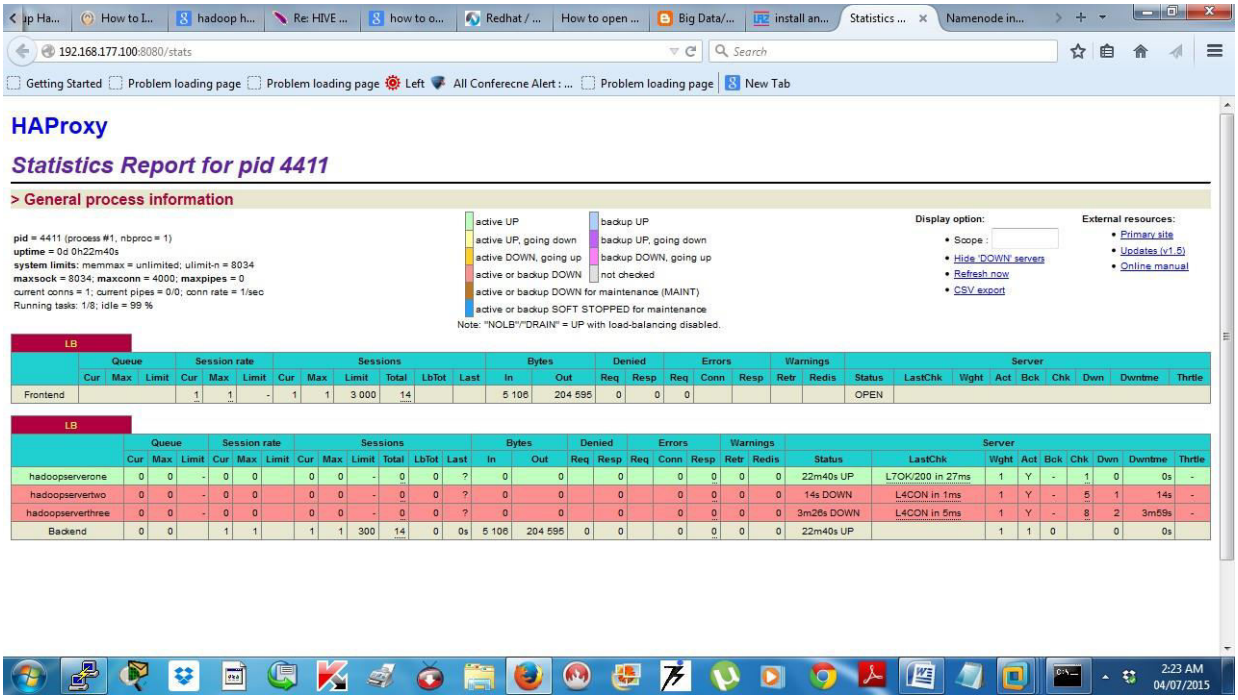

Figure 10: HAProxy rescheduled job on the first Server.

The above figure 9 shows HAProxy rescheduled job the first Server, when both two and server is not working

Case 3: Stop the Services of Hadoop Server one, HAPROXY Server Shows Status as down server and access to Hadoop is stopped.

This experiment is performed for the extreme case when all the servers are failed. As a result, all the services of the Hadoop node server will be stopped.

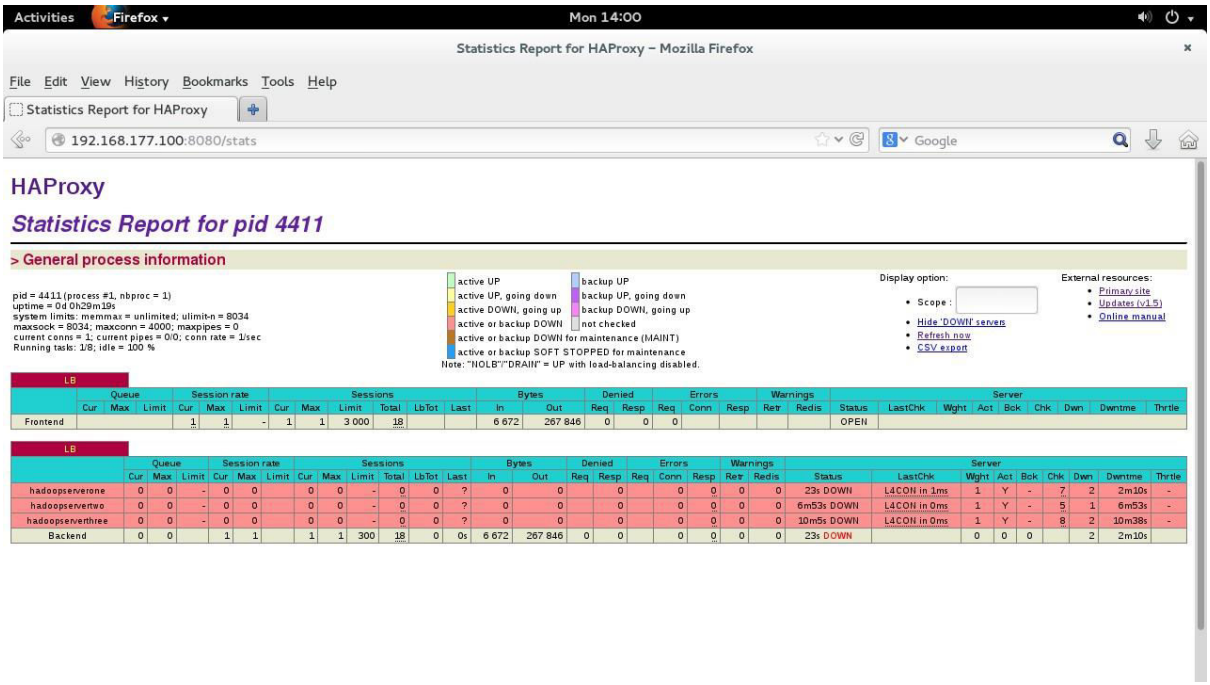

Figure 11: All Name node servers are down and all services HAProxy are stop.

The above figure 9, HAProxy shows all name node servers are down and all services HAProxy are stop.

Case 4: Start the Services of Hadoop Server one, HAProxy Server Shows Status as UP server and access to Hadoop is resumed.

The next case is tested when a node is up, HAProxy takes action immediately, puts all load to server one.

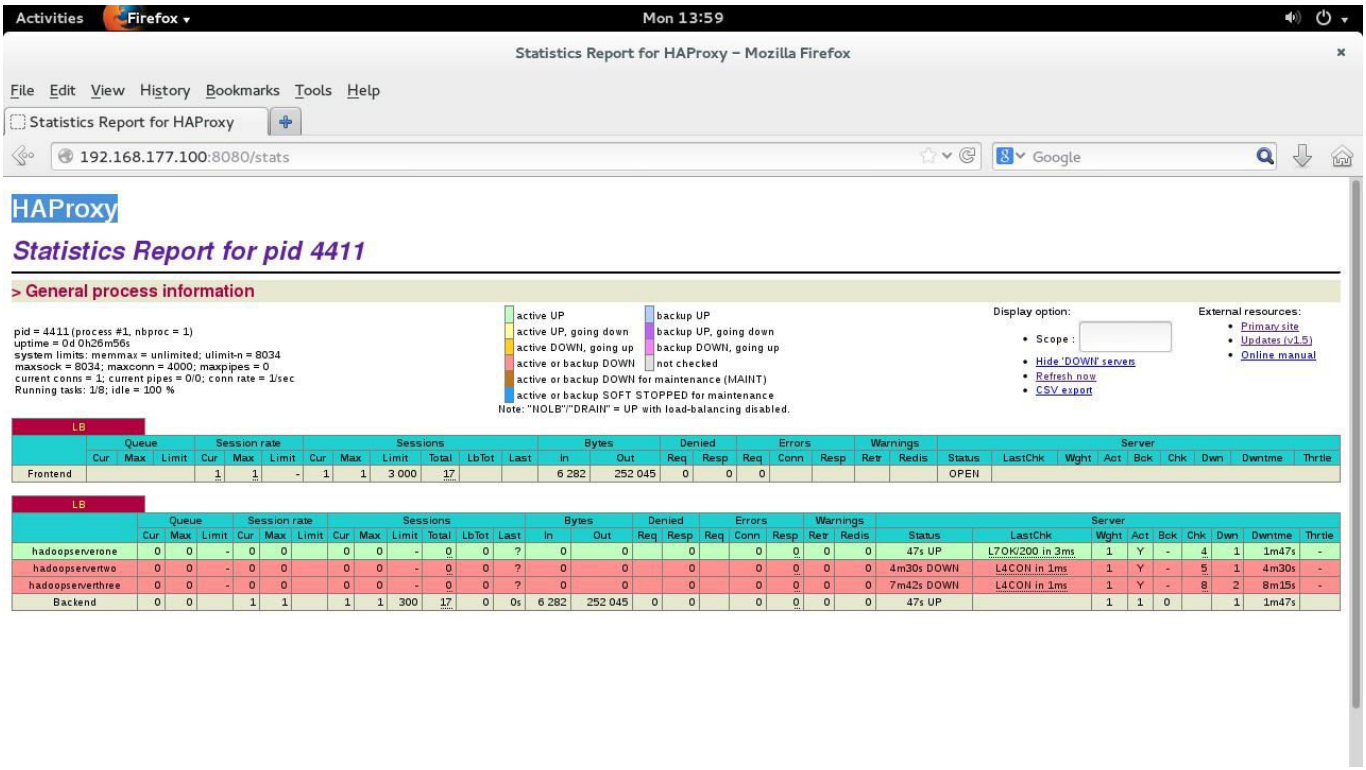

Figure 12: Show the status of servers when server one is up

In figure 12 shows when the server one resumes its services, HAProxy proxy immediately transfer control the first server. Hadoop services will be resumed.

Case 5: Start the Services of Hadoop Server one, HAPROXY Server Shows Status as UP server and is ready to handle fault tolerance.

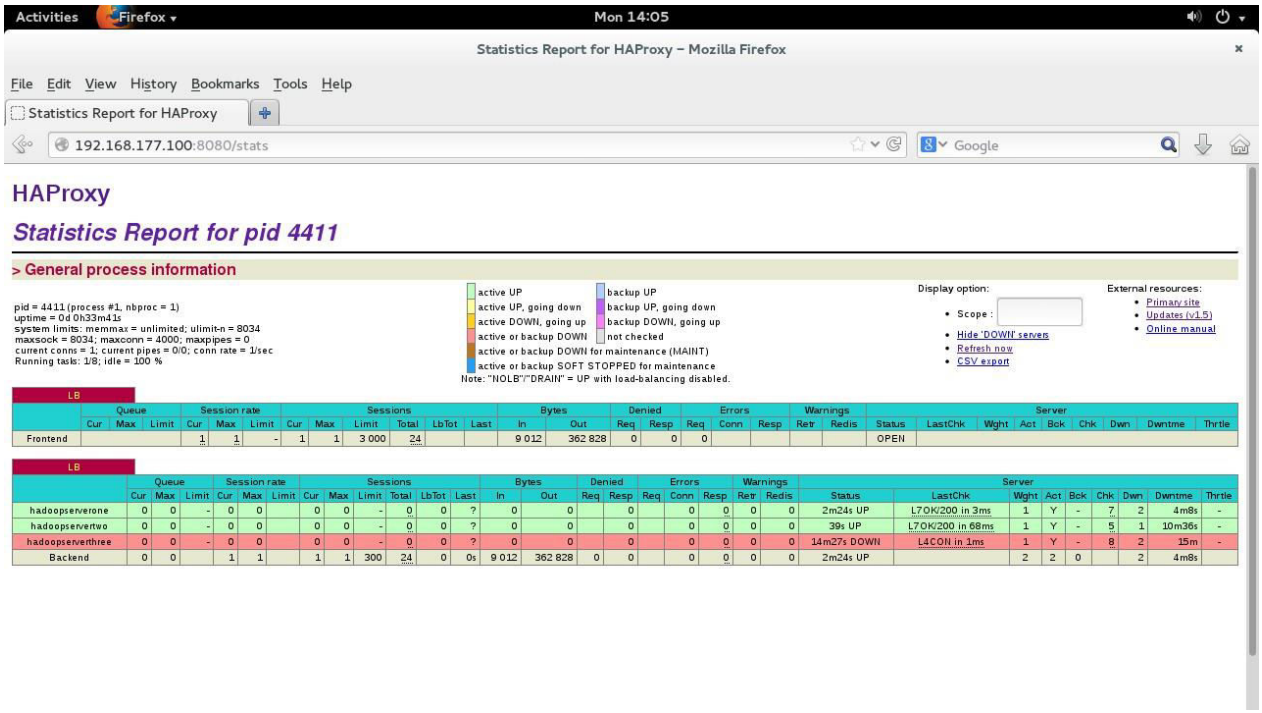

Figure 13: Shows the status of Hadoop servers.

Here two servers are running well. Only one Hadoop server is working. When the load of one node is high, then HAProxy will reschedule the load from one to another server.

Case 6: All servers are working fine.

In this case, the entire name node is working; HAProxy server is working on Round robin techniques. Their first name node hears all the requests in the cluster.

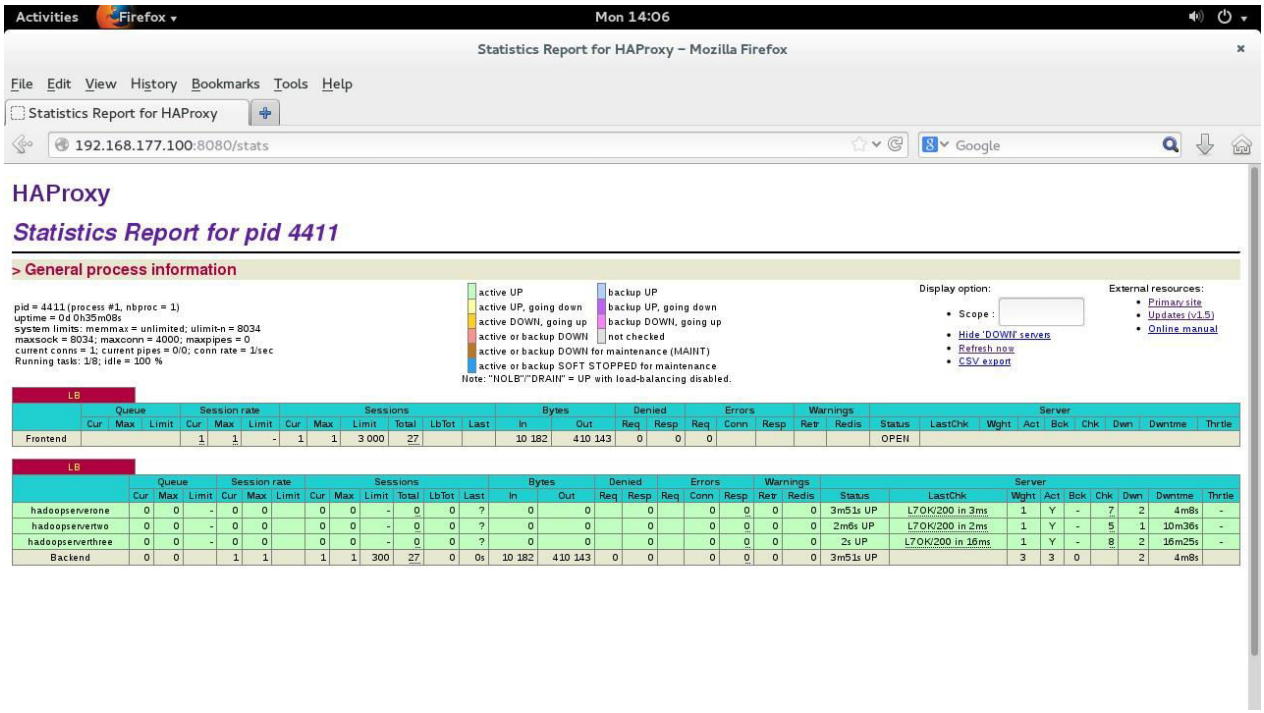

Figure 14: Show all name servers are working fine.

In this experiment, there are multiple experiments conducted to test the working of HAProxy Server. It also checks the HAProxy from different views. It shows the HAProxy migration technique which is based on the round robin scheme. In the experiment, HAProxy migrates one server to next server automatically.

The next experiment is conducted for testing all Name servers are mounted to NFS Server for accessing Hadoop System files.

5.3 The experiment to prepare the backup repository.

The experiment 2:

In this experiment, the NFS server crash handling of the Name Node server conduct test to prepare a common repository to the Hadoop files on the Linux server.

5.3.1 NFS Server Configuration and statistics

Server: NFS Server

IP address: 192.168.177.144 255.255.255.0

On this server, there is one NFS share directory, which too shares and accessible to the entire network. It stores the Hadoop name node root files back up with Synchronous mode. All Name Node servers Hadoop root directories are mounted with a share directory on NFS server. It provides the replica of root directories on all node servers on the NFS server.

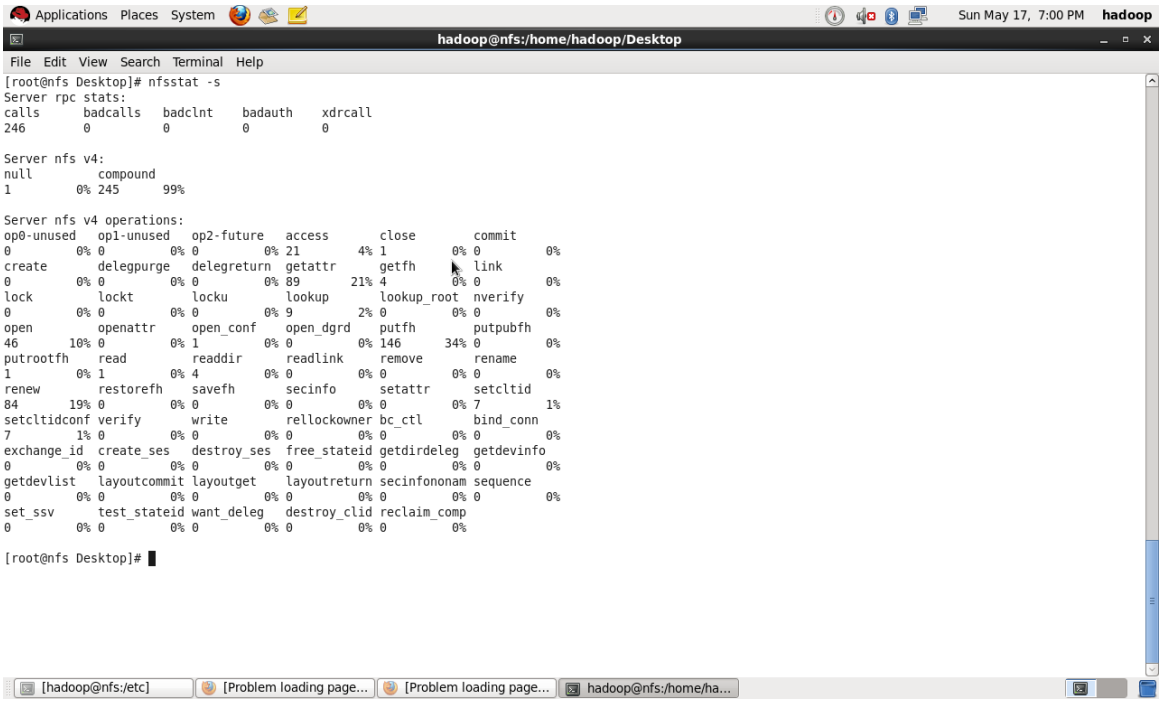

Figure 15 shows NFS Server Statistics

The above figure 15 shows, NFS Server statistics of hits and data transfer.

5.3.2 The NFS Clients Configuration and Statistics

The Hadoop directory on the name node 1 is mounted with a share directory on NFS Server. The name node acts as a client of NFS Server. The output of configuration and status of NFS clients is given below

5.3.2.1 Output of configuration and execution of commands on Namenode1

An IP Address: 192.168.177.103 255.255.255.0

```
$ mount -t 192.168.177.144:/share /Hadoop
[hadoop@localhost ~]$ cd hadoop
[hadoop@localhost hadoop]$ ls
bin  include  libexec     logs        README.txt  share
etc  lib      LICENSE.txt  NOTICE.txt  sbin
[hadoop@localhost hadoop]$ showmount -e 192.168.177.144
Export list for 192.168.177.144:
/share *
[hadoop@localhost hadoop]$ ls -l
total 56
drwxr-xr-x. 2 hadoop hadoop  4096 Nov 13  2014 bin
drwxr-xr-x. 3 hadoop hadoop  4096 Nov 13  2014 etc
drwxr-xr-x. 2 hadoop hadoop  4096 Nov 13  2014 include
drwxr-xr-x. 3 hadoop hadoop  4096 Nov 13  2014 lib
drwxr-xr-x. 2 hadoop hadoop  4096 Nov 13  2014 libexec
-rw-r--r--. 1 hadoop hadoop 15429 Nov 13  2014 LICENSE.txt
drwxrwxr-x. 3 hadoop hadoop  4096 Apr  6 14:07 logs
rw-r--r--. 1 hadoop hadoop   101 Nov 13  2014 NOTICE.txt
-rw-r--r--. 1 hadoop hadoop  1366 Nov 13  2014 README.txt
drwxr-xr-x. 2 hadoop hadoop  4096 Nov 13  2014 sbin
drwxr-xr-x. 4 hadoop hadoop  4096 Nov 13  2014 share
[hadoop@localhost hadoop]$
```

5.3.2.2 Output of configuration and execution of commands on Namenode2

Here a Hadoop root directory on the name node 2 is mounted with a share directory on NFS Server. The name node acts as the client of NFS Server. The output of configuration and status of NFS clients is given below

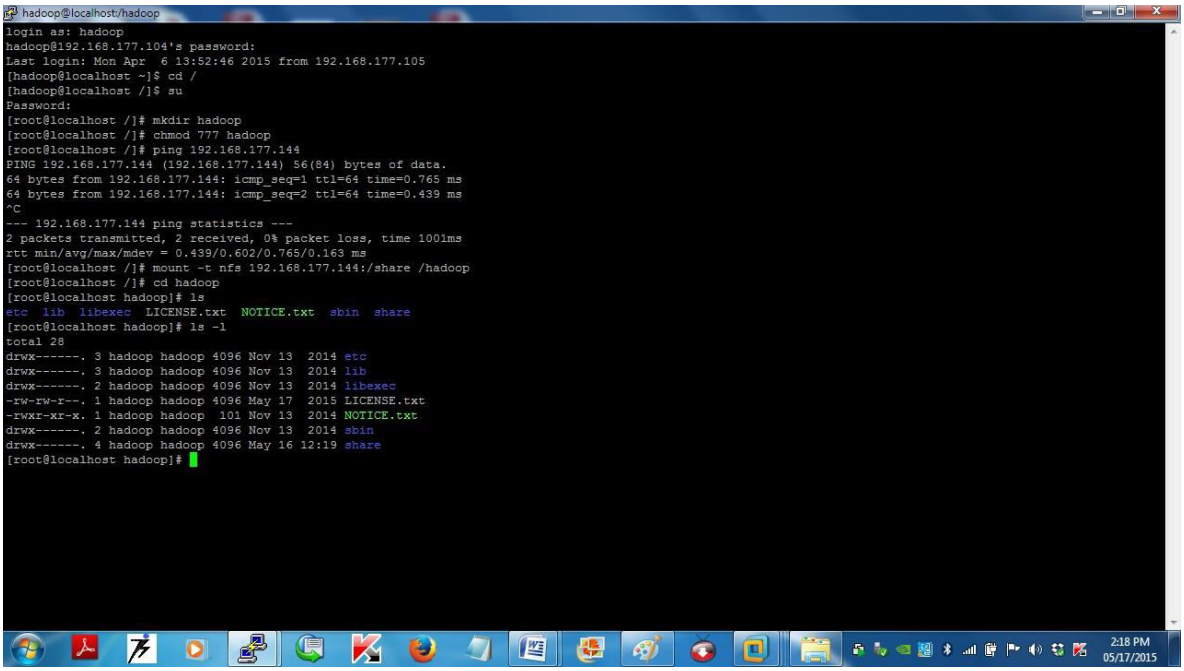

Figure 16: The Output of commands used for NFS Client on the Name Node 2

Name Node 16 Configuration:

IP Address: 192.168.177.104 Subnet Mask 255.255.255.0

Mount -t 192.168.177.144:/share /Hadoop

Output of execution of commands are given below

Login as: hadoop

hadoop@192.168.177.104's password:

Last login: Mon Apr 6 13:52:46 2015 from 192.168.177.105

[hadoop@localhost ~]$ cd /

[hadoop@localhost /]$ su

Password:

[root@localhost /]# mkdir hadoop

[root@localhost /]# chmod 777 hadoop

[root@localhost /]# ping 192.168.177.144

PING 192.168.177.144 (192.168.177.144) 56(84) bytes of data.

```
64 bytes from 192.168.177.144: icmp_seq=1 ttl=64 time=0.765 ms
64 bytes from 192.168.177.144: icmp_seq=2 ttl=64 time=0.439 ms
^C
--- 192.168.177.144 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.439/0.602/0.765/0.163 ms
[root@localhost /]# mount -t nfs 192.168.177.144:/share /hadoop
[root@localhost /]# cd hadoop
[root@localhost hadoop]# ls
etc  lib  libexec  LICENSE.txt  NOTICE.txt  sbin  share
[root@localhost hadoop]# ls -l
total 28
drwx------. 3 hadoop hadoop 4096 Nov 13  2014 etc
drwx------. 3 hadoop hadoop 4096 Nov 13  2014 lib
drwx------. 2 hadoop hadoop 4096 Nov 13  2014 libexec
-rw-rw-r--  1 hadoop hadoop 4096 May 17  2015 LICENSE.txt
-rwxr-xr-x. 1 hadoop hadoop  101 Nov 13  2014 NOTICE.txt
drwx------. 2 hadoop hadoop 4096 Nov 13  2014 sbin
drwx------. 4 hadoop hadoop 4096 May 16 12:19 share
```

5.3.2.3 Output of configuration and execution of commands on the Name Node 3

Name Node 3

Configuration of NFS client on Name node 3

IP Address : 192.168.177.105 Subnet Mask 255.255.255.0

Output of execution of commands are given below

mount -t 192.168.177.144:/share /hadoop

```
[hadoop@localhost ~]$ cd hadoop

[hadoop@localhost hadoop]$ ls

bin  include  libexec     logs        README.txt  share

etc  lib      LICENSE.txt  NOTICE.txt  sbin

[hadoop@localhost hadoop]$ showmount -e 192.168.177.144

Export list for 192.168.177.144:

/share *

[hadoop@localhost hadoop]$ ls -l

total 56

drwxr-xr-x. 2 hadoop hadoop  4096 Nov 13  2014 bin

drwxr-xr-x. 3 hadoop hadoop  4096 Nov 13  2014 etc

drwxr-xr-x. 2 hadoop hadoop  4096 Nov 13  2014 include

drwxr-xr-x. 3 hadoop hadoop  4096 Nov 13  2014 lib

drwxr-xr-x. 2 hadoop hadoop  4096 Nov 13  2014 libexec

-rw-r--r--. 1 hadoop hadoop 15429 Nov 13  2014 LICENSE.txt
```

In this experiment, all the root directories of name nodes are mounted on a single share directory on the NFS Server. This technique provides same data and configuration on all the Hadoop servers. The NFS Share is used as synchronous mode. When one Hadoop Server data is updated, then it will automatically update data on other Name Nodes.

Chapter 6

Conclusion and Future Scope

This chapter discusses the conclusions of work presented in this thesis. The chapter ends with a discussion of the future direction which thesis work can take.

6.1. The Conclusion

The thesis discusses an evolution of the Distributed File System in Cloud Computing. A fault tolerance framework has been proposed and developed to implement fault tolerance for handling a crash failure of Master Name Node Server of Apache Hadoop. The proposed framework has used Job Migration and data replication techniques tolerating the crash of Master Name Node Server. The solution has provided an automatic and transparent fault tolerance. Various tools have been used in the framework capable of recovering from the faults in few milliseconds. High availability and reliability has been achieved in this framework without promoting Secondary Name Node. The proposed framework has been evaluated using prototype HAProxy Server, various Name Nodes and NFS Server in the cluster. HAProxy has been used for monitoring, load balancing and job migration of Name Nodes. NFS Server has also been used for replication of data of the Hadoop in the cluster. The experimental results show achievement of high availability of Master Name Node Server without suffering loss of data and break of services.

6.2 Thesis Contribution

- The Virtualized Hybrid Cloud Architecture is used for implementing the framework.
- Various tools propose the real time fault tolerance monitoring.
- A prototype is designed, implemented and analyzed using the proposed framework.
- An effort to provide a common replica of all name nodes at the NFS share directory service of Linux.
- An effort to provide the high availability of services on the cluster.

6.3 Future Scope

- Some future extensions can also be possible by considering VM Server programming of automatic cloning of VM Machines.
- It can extend for the high performance computing application.
- It can be extended on Google Cloud Platform for huge transactions.

References

[1]. Peter Mell, Timothy Grance ―The NIST Definition of Cloud Computing‖, NIST Special Publication 800-145, September, 2011.

[2]. VMware Workstation from Wikipedia, the free encyclopedia, http:// http://en.wikipedia.org/wiki/VMware_Workstation

[3]. "HAProxy Documentation Manual "[online] Available: http://haproxy.1wt.eu [Accessed: 22 March 2013]

[4] Fedora Wikipedia http://en.wikipedia.org/wiki/Fedora_%28operating_system%29

[5] "HDFS (Hadoop distributed file system) architecture", [online] Available: http://hadoop.apache.org/common/docs/current/hdfs/design.html2009, Accessed. [13 April 2013]

[6] Chapter 14 Network files system http://www.oreilly.com/openbook/linag2/book/ch14.html

[7] R. Buyya and M. Murshed, "Grasim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. Concurrency and Computation: Practice and Experience‖, Wiley Press, 14(13- 15), Nov.-Dec., 2002.

[8]. Sheheryar Malik, Fabrice Huet, ―Adaptive Fault Tolerance in Real Time Cloud Computing‖, IEEE World Congress on Services, 2011.

[9]. Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing Anju Bala, Thapar University.

[10]. Fault Tolerance Techniques and Comparative Implementation in Cloud Computing , Prasenjit Kumar Patra, Lovely Professional University, Punjab

[11]. CLOUD COMPUTING Principles and Paradigms Edited by Rajkumar Buyya The University of Melbourne and Manjra soft Pty Ltd., Australia James Bromberg The University of Melbourne, Australia Andrzej Goscinski Deakin University, Australia.\

[12]. Fault Tolerance and Resilience in Cloud Computing Environments Ravi Jhawar and Vincenzo Piuri.

[13]. Emerging Cloud Computing Paradigm Abu Sarwar Zamani 1, Md. Mobin Akhtar 2 and Sultan Ahmad 3 1 College of Science, Shaqra University, Al-Quwaiyyah, Kingdom of

Saudi Arabia 2 Al-Qiayaha Community College, Shaqra University Kingdom of Saudi Arabia 3 College of Computer Engineering & Sciences, Al-Kharj University Al-Kharj, Kingdom of Saudi Arabi.

[14]. Hadoop Map/Reduce Tutorial

[15]. Making Cloud Intermediate Data Fault-Tolerant Steven Y. KO Imranul Hoque† Brian Cho† Indranil Gupta.

[16]. From Crash Fault-Tolerance to Arbitrary-Fault Tolerance: Towards a Modular Approach Roberto BALDONI Jean-Michel HELARY Michel RAYNAL.

[17]. Checkpoint-based Fault-tolerant Infrastructure for Virtualized Service Providers ´Inigo Goiri, Ferran Juli`a, Jordi Guitart, and Jordi Torres Barcelona SuperComputing Center and Technical University of Catalonia, Jordi Girona 31, 08034 Barcelona, Spain {Igoiri, fjulia, jguitart, Torres}@ac.upc.edu

[18]. Improving Fault Tolerance through Crash Recovery Tsozen Yeh, Department of Computer Science and Information Engineering Fu Jen Catholic University New Taipei City 24205, Taiwan eh@csie.fju.edu.tw Weian Cheng Department of Computer Science and Information Engineering Fu Jen Catholic University New Taipei City 24205, Taiwan n96@csie.fju.edu.tw

[19]. On the Performance of Byzantine Fault-Tolerant MapReducePedro Costa, Marcelo Pasin, Alysson Neves Bessani, and Miguel P. Correia.

[20] Benchmarking Dependability of Map Reduce Systems Amit Sangroya INRIA - LIG Grenoble, France Amit.Sangroya@inria.fr Dami´an Serrano INRIA - LIG Grenoble, France Damian.Serrano@inria.fr Sara Bouchenak University of Grenoble - LIG - INRIA Grenoble, France Sara.Bouchenak@inria.fr

[21] High Performance RDMA-based Design of HDFS over InfiniBand* N. S. Islam1, M. W. Rahman1, J. Jose1, R. Rajachandrasekar1, H. Wang1, H. Subramoni1, C. Murthy2, and D. K. Panda1 1 Department of Computer Science and Engineering, 2 IBM T.J Watson Research Center the Ohio State University Yorktown Heights, NY {islamn, rahmanmd, Jose, rajachan, wangh, subramon, panda} {chet} @cse.ohio-state.edu @watson.ibm.com

[22] Apt Store: Dynamic Storage Management for Hadoop, Krish K. R. †, Aleksandr Khasymski†, Ali R. Butt†, Sameer Tiwari‡, Milind Bhandarkar‡ †Virginia Tech, ‡Greenplum {kris, khasymskia,butta}@cs.vt.edu, {Sameer.Tiwar, Milind.Bhandarkar}@emc.com

[23] Sheheryar MalikandFabriceHuet “Adaptive Fault Tolerance in Real Time Cloud Computing” 2011 IEEE World Congress on Service

List of Publication

1. Rajeev Kapoor, Ms. Anju Bala ," Fault Tolerance Against Crash Failures of Name Nodes of Hadoop By Using Load Balancing Techniques " in National Conference On Interdisciplinary Research In Science and Technology (17th April 2015) ISBN : 978-81-924212-9-2 at Dept of Electronics & Comm. and Dept of Computer Sc. Lingaya's GVKS Institute of Management & Technology, Faridabad.
   International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 8, Number 1 (2015) © International Research Publication House • http://www.irphouse.com
   http://www.irphouse.com/volume%20special/ijertv8n1spl.htm