A Proposed Framework for Fault Prediction and Monitoring in Cloud Environment

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

Master of Engineering
in
Software Engineering

Submitted By
Dhananjaya Gupt
(Roll No. 801131008)

Under the supervision of:
Ms. Anju Bala
Assistant Professor
C.S.E.D, Thapar University

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2013

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*A Proposed Framework for Fault Prediction and Monitoring in Cloud Environment* ", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software 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 *Ms. 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.

(Dhananjaya Gupt)

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

(Ms. Anju Bala)
Assistant Professor
Computer Science and Engineering Department,
Thapar University, Patiala

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

i

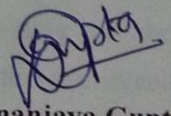
Acknowledgement

No volume of words is enough to express my gratitude towards my guide **Ms. Anju Bala**, Department of Computer Science & Engineering, Thapar University, Patiala, who has been very concerned and has aided for all the materials essentials for the preparation of this thesis report. She has helped me to explore this vast topic in an organized manner and provided me all the ideas on how to work towards a research-oriented venture.

I am also thankful to **Dr. Maninder Singh**, Head of Computer Science & Engineering Department and **Mr. Karun Verma**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there at the need of hour and provided with all the help and facilities, which I required, for the completion of my thesis work.

Most importantly, I would like to thank my parents and the almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.



Dhananjaya Gupt

(801131008)

Abstract

With the enormous growth of internet and its users, Cloud computing, with its great potentials in low cost and on-demand services, has become a promising computing platform for both commercial and non-commercial computation clients. In this thesis various fault tolerance techniques have been discussed to improve performance and availability of applications running in cloud environment.

In cloud computing, users do not have any information about the physical resources like server location, server capacity etc. whenever any fault or failure occurs in any physical resource, vendor has to deal with the problem to provide uninterrupted service to end user. Reliability, availability in Cloud computing are critical requirements to ensure correct and continuous system operation also in the presence of failures. Therefore there is need to design a Fault tolerance framework using fault prediction and monitoring tools.

The solution provided in this thesis make use of a software tool known as HAProxy, Hadoop and Nagios that offers high availability and can handle failures in the framework of virtual machines. Using these tools, a Fault prediction and monitoring Framework has been proposed and a prototype is implemented in Linux by using two web server applications that may comprise of faults. HAProxy balances load between these servers and migrates request between server on failure. Also data is stored in Hadoop that provides data replication at multiple nodes. Nagios provides a Fault Prediction mechanism to predict fault in Hadoop before complete cluster goes down. The complete framework provides autonomic and transparent fault tolerance capability to cloud applications. The experimental results show that framework comprising of above mentioned tools can make applications to recover from these faults in a few milliseconds thus increasing system availability and reliability.

Table of Contents

Chapter 1
Introduction

In this chapter the basic introduction about cloud computing, its background, model and architecture details are described in detail. The benefits, application and important issues have also been discussed. The end of this chapter gives the structure of thesis.

1.1 Background

The underlying concept of cloud computing dates back to the mainframe days of 1960's when the idea of utility computing was coined.MIT computer scientist John McCarthy depicted that "computation may someday be organized as a public utility". Utility computing became a sort of business for companies such as IBM. IBM saw the potential for enormous profit to be made in this type of business and started providing computing services. In the mid 1990s, Grid computing grew up with the idea of linking the number of computers to increase scalability and availability. The grid specifically refers leverage of computers for particular application while cloud computing leverages the multiple resources along with the computational resources to provide the services-to-the-end-user.

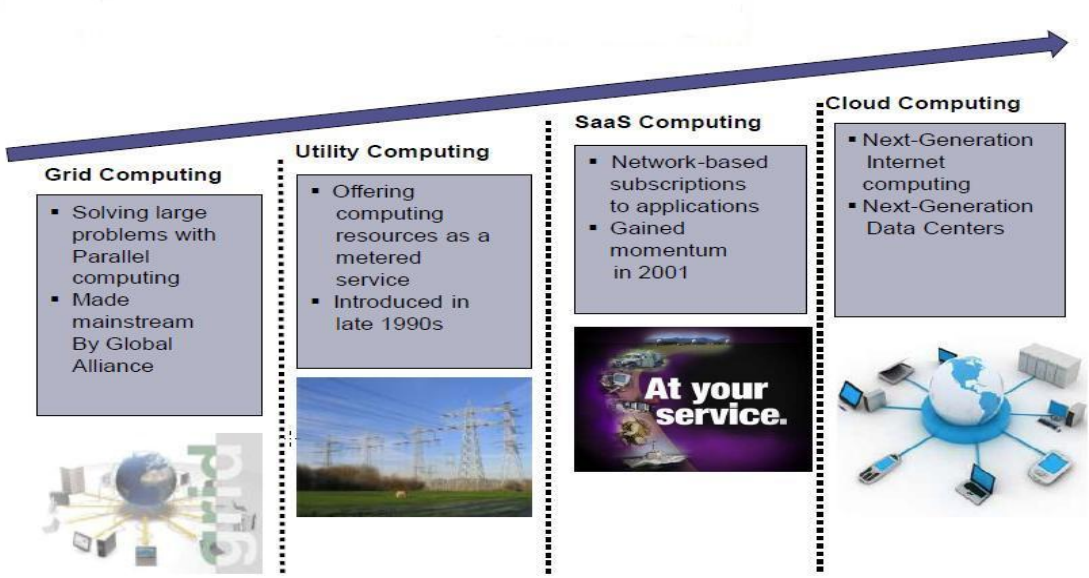


Figure 1.1: Evolution of Cloud from Grid Computing [1]

1.2 Introduction to Cloud Computing

The National Institute of Standards and Technology (NIST) defines cloud computing as "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". This cloud model is composed of five essential characteristics, three service models, and four deployment models [2].

Cloud computing model offers several benefits that include an improved peak-load handling and dynamic resource provisioning without the need for setting up a new software or hardware infrastructure in every location [3]. Cloud computing improve services by optimizing the service location and throughput according to user's QoS needs. It provides higher reliability as providers can transparently migrate their services to other domains, thus enabling a highly resilient computing environment [4].

1.3 Cloud Computing Service Models

Cloud computing is emerging as a model in support of "everything-as-a-service" (XaaS). Service delivery in Cloud Computing comprises three different service models, namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

1.3.1 Infrastructure as a Service (IaaS):

IaaS is a service delivery model in which an organization is given control over the different resources and applications. These resources comprise of storage, hardware, servers, networking components, etc. On the lowest level of the infrastructure closest to the hardware two types of services are distinguished: Physical Resource Set (PRS) and Virtual Resource Set (VRS) services [12]. The PRS layer implementation is hardware dependent and therefore tied to a hardware vendor, whereas the VRS layer can be built on vendor independent hypervisor technology. Examples of PRS services are Emulab and iLO and VRS services include Amazon EC2, Eucalyptus, Tycoon, Nimbus, and OpenNebula [8]. Benefits of IaaS are:

- Systems managed by SLA should equate to fewer breaches
- Higher return on assets through higher utilization
- Reduced cost driven by Less hardware
- Less floor space from smaller hardware footprint
- Higher level of automation from fewer administrators
- Lower power consumption
- Able to match consumption to demand

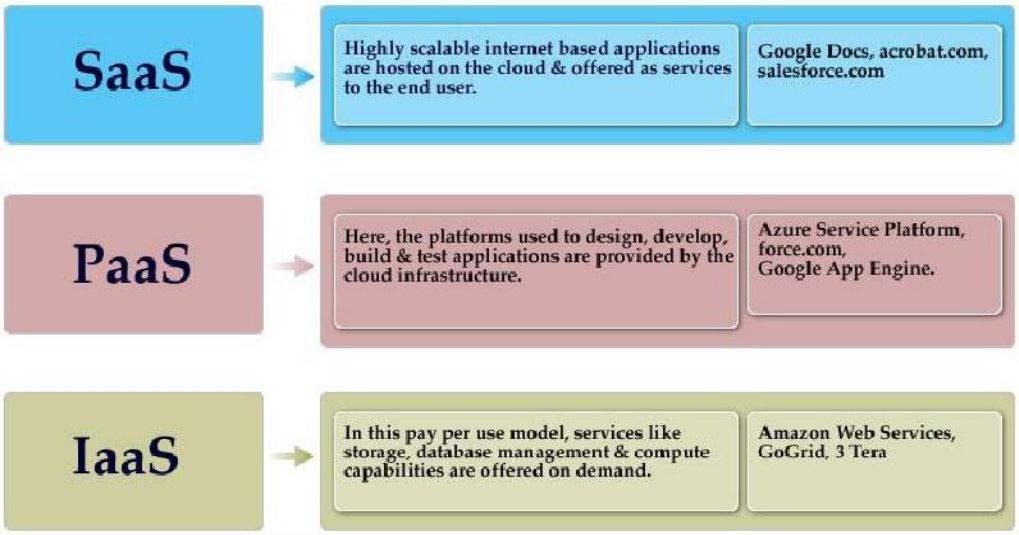


Figure 1.2: Cloud Computing Service Model [16]

1.3.2 Platform as a Service (PaaS):

Paas can be defined as a set of software and product development tools that allows developer to create applications on the provider's platform, i.e., applications can be build on the internet and executed on providers infrastructure[12]. The services are categorized into Programming Environments and Execution Environments. Example of the former is Sun's project Caroline and the Django framework, and examples of the latter are Google's App Engine, Joyent's Reasonably Smart and Microsoft's Azure [8]. Following are the benefits of PaaS:

- Pay-as-you-go for development, test, and production environments
- Enables developers to focus on application code

- Instant global platform
- Elimination of H/W dependencies and capacity concerns

1.3.3 Software as a Service (SaaS):

Saas is a software delivery model in which software and associated data are centrally hosted on the cloud. This is the most familiar and prolific cloud service. All the applications that run on the Cloud and provide a direct service to the customer are located in the SaaS layer [12]. The application developers can either use the PaaS layer to develop and run their applications or directly use the IaaS infrastructure. Here, Basic Application Services (BAS) and Composite Application Services (CAS) are distinguished. Examples of Basic Application Services are the OpenId and Google Maps services. In the Composite Application Service category the mash-up support systems with Open social are the prominent example allowing entire social networks like MySpace to be used as Basic Services [8]. Following are the benefits of SaaS**:**

- Speed
- Reduced up-front cost, potential for reduced lifetime cost
- Transfer of some/all support obligations
- Elimination of licensing risk
- Elimination of version compatibility
- Reduced hardware footprint

1.4 Cloud Deployment Models

In present cloud computing paradigm, clouds are categorized on the basis of the location where it is deployed [6].

1.4.1 Public cloud:

In Public cloud the computing infrastructure is hosted by the cloud vendor at the vendor's premises. The customer has no visibility and control over where the computing infrastructure is hosted [6]. The computing infrastructure is shared between any organizations. Public cloud benefits low investment hurdle: pay for what you use and good test/development environment for applications that scale to many servers. Amazon

Elastic Cloud Compute (EC2), Google App Engine, Blue Cloud by IBM are few examples of Public cloud.

1.4.2 Private cloud:

The computing infrastructure is dedicated to a particular organization and not shared with other organizations. Some experts consider that private clouds are not real examples of cloud computing. Private clouds are more expensive and more secure when compared to public clouds [6]. Private clouds are of two types: On-premise private clouds and externally used by one organization, but are hosted by a third party specializing in cloud infrastructure. Externally hosted private clouds are cheaper than On-premise private clouds. Private cloud benefits fewer security concerns as existing data center security stays in place and IT organization retains control over data center. Examples of private cloud are OpenStack, CloudStack etc.

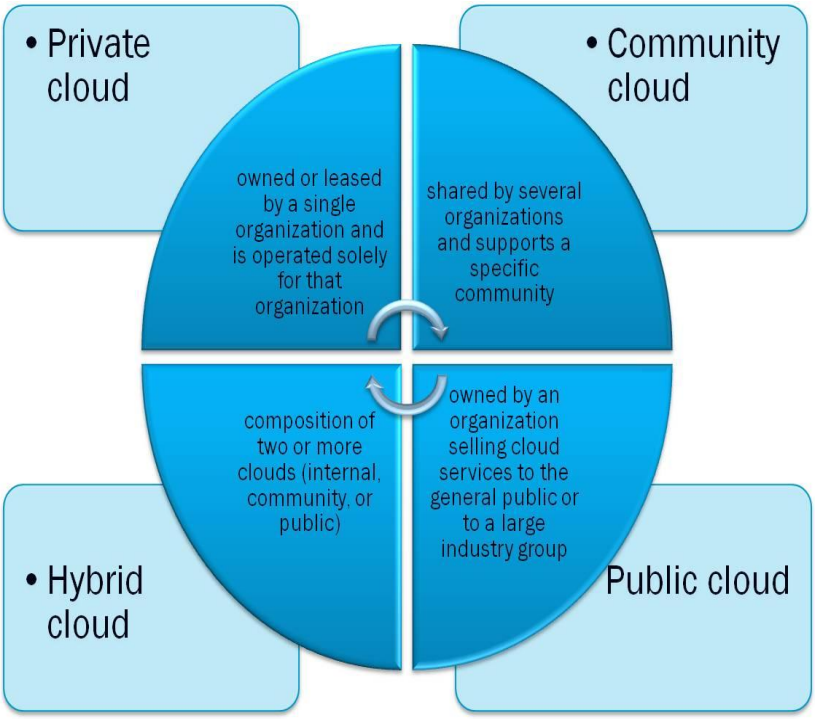


Figure 1.3: Cloud Deployment Model [20]

1.4.3 Hybrid cloud:

Organizations may host critical applications on private clouds and applications with relatively less security concerns on the public cloud. The usage of both private and public clouds together is called hybrid cloud [6]. A related term is Cloud Bursting. In Cloud bursting organization use their own computing infrastructure for normal usage, but access the cloud for high/peak load requirements. This ensures that a sudden increase in computing requirement is handled gracefully.

1.4.4 Community cloud:

A community cloud is established where several companies share similar requirements and look to gain the benefits from sharing the same resources. One example comes from the Life Sciences area where electronic data capture (EDC) vendors offer their clinical trials tools via cloud computing [41].

1.5 Layered Architecture of Cloud

Clouds can be built on top of many existing protocols such as Web Services (WSDL, SOAP), and some advanced Web 2.0 technologies such as REST, RSS, AJAX, etc. In fact, behind the cover, it is possible for Clouds to be implemented over existing Grid technologies leveraging more than a decade of community efforts in standardization, security, resource management, and virtualization support [8]. Figure 1.4 shows the layered design of service-oriented Cloud computing architecture. Physical Cloud resources along with core middleware capabilities form the basis for delivering IaaS. The user-level middleware aims at providing PaaS capabilities. The top layer focuses on application services (SaaS) by making use of services provided by the lower layer services. PaaS/SaaS services are often developed and provided by 3rd party service providers, who are different from IaaS providers [9].

- **User-Level Middleware:** This layer includes the software frameworks such as Web 2.0 Interfaces (Ajax, IBM Workplace) that help developers in creating rich, cost effecting User-interfaces for browser-based applications .The layer also provide the

programming environments and composition tools that ease the creation, deployment, and execution of applications in Clouds.

- **Core Middleware:** This layer implements the platform level services that provide runtime environment enabling Cloud computing capabilities to application services built using User-Level Middleware. Core services at this layer includes Dynamic SLA Management, Accounting, Billing, Execution monitoring and management, and Pricing. The well- known examples of services operating at this layer are Amazon EC2, Google App Engine, and Aneka [5].

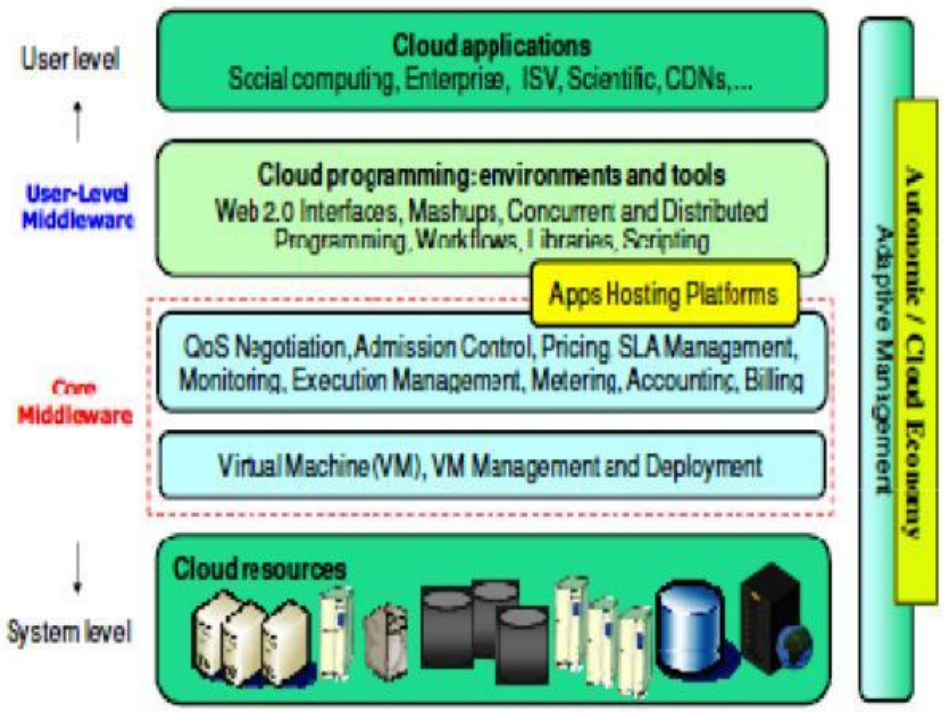


Figure 1.4: Layered Cloud Computing Architecture [15]

- **System Level:** The computing power in Cloud computing environments is supplied by a collection of data centres, which are typically installed with hundreds to thousands of servers [14]. At the System Level layer there exist massive physical resources (storage servers and application servers) that power the data centres. These servers are transparently managed by the higher level virtualization services and

toolkits that allow sharing of their capacity among virtual instances of servers [13]. These VMs are isolated from each other, which aid in achieving fault tolerant behaviour and isolated security context.

1.6 Benefits of Cloud Computing

Cloud computing offers a number of benefits, including the potential for:

- **Rapid scalability and deployment capabilities**: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time [39].
- **Capacity for on-demand infrastructure and computational power**: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider [39].
- **Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter) [38].
- **Measured Service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service [39].
- **Access to Top-End IT Capabilities:** Particularly for smaller organizations, cloud computing can allow access to higher-caliber hardware, software, and staff than they can attract and/or afford themselves [38].
- **Cloud brokers**: Cloud services brokers simplify an organization's transition to the cloud by helping to overcome specific security, privacy and compliance issues and

helping achieve interoperability across multiple public clouds, private clouds and in-house IT infrastructure [40].

- **Market barrier reduction:** cloud computing services reduce IT barriers to market entry, enabling far more start-ups to emerge with much lower infrastructure costs than were necessary pre-cloud. This increases innovation in and of itself and also spurs larger organizations to innovate more rapidly [40].

1.7 Issues in Cloud computing

This section discusses about the important issues and their impact on cloud computing environment. These issues need to be resolved to achieve stability in real time applications running in cloud environment.

1.7.1 Security and Privacy issues

There are numerous security issues for cloud computing as it encompasses many technologies including networks, databases, operating systems, virtualization, resource scheduling, transaction management, load balancing, con- currency control and memory management [42]. Many organizations are uncomfortable with the idea of storing their data and applications on systems they do not control. Migrating workloads to a shared infrastructure increases the potential for unauthorized access and exposure [40]. Consistency around authentication, identity management, compliance, and access technologies will become increasingly important. To reassure their customers, cloud providers must offer a high degree of transparency into their operations. Before making a choice of cloud vendors, users should ask the vendors for seven specific safety issues: Privileged user access, regulatory compliance, data location, data segregation, recovery, investigative support and long-term viability [4].

1.7.2 Data management issues

One of the major issues with clouds is that the owner of the data may not have control of where the data is placed. If one wants to exploit the benefits of using cloud computing, one must also utilize the resource allocation and scheduling provided by clouds. Therefore, we need to safeguard the data in the midst of untrusted processes [42]. Cloud does not differentiate between a sensitive data from a common data thus enabling anyone

to access those sensitive data's. Most of the cloud Vendors instead of acquiring a server tries to lease a server from other service providers because they are cost affective and flexible for operation. The customer doesn't know about those things, there is a high possibility that the data can be stolen from the external server by a malicious user. Data loss is a very serious problem in Cloud computing. When it comes to location of the data nothing is transparent even the customer doesn't know where his own data's are located. The Vendor does not reveal where all the data's are stored. The Data's won't even be in the same country of the Customer, it might be located anywhere in the world.

1.7.3 Availability issues

Availability in cloud computing cannot be ensured as user cannot simply assume, however, that any cloud based service from any Cloud Service Provider (CSP) will always be there every time they need it. Although the business models of the largest CSPs involve multiple data centers in different geographies, spreading and balancing workloads among them, many do not. Often, CSPs have poor regional or global coverage that might affect a customer's experience [40].

1.7.4 Performance issues

In cloud computing, it is difficult to ensure performance as application performance will be impacted as services are moved into the cloud. The big question to be answered is how development, performance and test engineers can ensure that their applications will be resilient and meet performance expectations in the unknown cloud environment [40].

1.7.5 Fault tolerance issues

Providing fault tolerance requires careful consideration and analysis not only because of their complexity and inter-dependability but. Cloud infrastructure consists of different types of hardware/software technologies, which are provided by multiple, and most likely competing vendors [27]. The fault tolerance implementation requires complex communications at different stages across various cloud entities. Cloud infrastructure is not hosted at a single data center that is located at a specific location; it is rather the opposite, as most likely it is distributed across distant data centers [27]. This factor has a major impact on decisions being made by fault tolerance mechanisms for several reasons;

for example, the distance and the communication medium between distant data centers will have an impact on data transfer speed.

1.8 Structure of Thesis

The chapters in this thesis are organized as follows –

Chapter 2 – This chapter describes in detail the literature survey.

Chapter 3 – This chapter describes the problem statement of the thesis. It gives the gap analysis and requirement analysis.

Chapter 4 – This chapter describes the solution of problem, design of solution, system Framework and description of tools used for implementation.

Chapter 5 – This chapter gives the Implementation and Experimental results –Web application snapshots, Hadoop snapshots, HAProxy snapshots, Snapshots of HAProxy statistics, Nagios snapshots.

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

Chapter 2
Literature Review

Previous chapter provides 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 Introduction to Fault Tolerance

Fault Tolerance refers to an approach to system design that allows a system to continue performing even when one of its components fails. If not possible then fault tolerance solutions may allow a system to continue operating at reduced capacity rather than shutting down completely following a failure [24]. Since about 1970, the field of fault-tolerant computing has been rapidly growing.

2.2 Fault Tolerant Computing

Fault Tolerant computing has been defined as "the ability to execute specified algorithms correctly regardless of hardware failures, total system flaws, or program fallacies" [18]. A fault-tolerant system may be able to tolerate one or more fault-types including transient, intermittent or permanent hardware faults, software and hardware design errors, operator errors, externally induced upsets or physical damage. An extensive methodology has been developed in this field over the past thirty years, and a number of fault-tolerant machines have been developed. Most of mechanism dealing with random hardware faults, while a smaller number deal with software, design and operator faults to varying degrees.

Fault tolerance and dependable systems research covers a wide spectrum of applications ranging across embedded real-time systems, commercial transaction systems, transportation systems, and military/space systems etc. The supporting research includes system architecture, design techniques, coding theory, testing, validation, proof of correctness, modeling, software reliability, operating systems, parallel processing, and real-time processing [18]. These areas often involve widely diverse core expertise ranging from formal logic, mathematics, graph theory, hardware design and software engineering.

2.3 Importance of Fault tolerance in cloud environment

With the increasing demand and benefits of cloud computing infrastructure, real time computing can be performed on cloud infrastructure. A real time system can take advantage of intensive computing capabilities and scalable virtualized environment of cloud computing to execute real time tasks. In most of the real time cloud applications, processing is done on remote cloud computing nodes. So there are more chances of errors, due to the undetermined latency and loose control over computing node [14]. On the other side, most of the real time systems are also safety critical and should be highly reliable so they require a higher level of fault tolerance. These systems require working properly to avoid failure, which can cause financial loss as well as casualties [26]. So there is an increased requirement for fault tolerance to achieve reliability for the real time computing on cloud infrastructure. The basic mechanism to achieve the fault tolerance is replication.

2.4 Management of Fault tolerance

Fault can be introduced at any stage in the system. There are three levels in cloud computing where FT management can be deployed. It can be either at Application level, virtual machine or hardware [44].

i. **Application FT:** Application failures are detectable only at the customer level. The failure detection policy depends on the application. For each application, the customer deploys in the cloud special software components called sensors, which monitor the aliveness of the application. According to this monitoring, a sensor may trigger the execution of a procedure for repairing the application when it is considered as malfunctioning [44].

ii. **Virtual Machine FT:** VM failures can be detected and repaired by the two cloud participants. Customers implement VM FT by deploying sensors in the cloud, which monitor VM state during their lifetime. In this case, it is not recommended to give to a single sensor the responsibility of probing one VM because a failure of the VM hosting this sensor would compromise the repair mechanism. The repair of the failed VM is organized as follows [44]:

- Customer level requests the cloud to free the failed VM.
- It allocates a new VM.
- It deploys and starts the servers that were running on the failed VM.
- It restores the state of these servers in case of statefull servers.

One disadvantage of this solution is that each customer must implement his own VM monitoring system, which leads to complexity and network resource wasting, while VM monitoring can be factorized and implemented by the cloud provider.

At the cloud provider level, an exclusive VM FT technique can be implemented. With a direct access to VM hypervisors, the cloud provider is more likely to implement VM FT. First, such an implementation decreases the number of VM sensors (and their associated communication) as they are integrated in hypervisors. Actually, a single sensor per physical machine can monitor all the VMs hosted on this machine. Secondly, through the hypervisor, the provider can collect more detailed information about VM status. This information allows him to implement more accurate FT solutions according to VM status.

iii. **Physical Machine FT:** Hardware failures are more difficult to take into account in the cloud because they trigger failures at the other levels (VM and application). At the customer level, it is impossible to detect a physical machine failure. At this level, a physical machine failure is perceived as multiple VM failure. The customer level only sees VMs and even if VM monitoring sensors are deployed in the cloud, they may not be aware of a hardware failure if they are all deployed on the same physical machine. The customer would need to implement constraints regarding the physical location of sensors, which is a form of collaboration between the customer and the cloud. At cloud provider level, hardware FT is implemented with a monitoring system composed of sensors deployed on different physical machines. For repair, the provider will start on a new machine (or several machines according to its capacities) the same number of VMs, which were hosted on the failed machine. In addition, all VM states must be saved by checkpointing so that VM restoration is possible [44].

2.5 Fault Tolerance Techniques

Fault tolerance techniques are designed to allow a system to tolerate software faults that remain in the system after its development. Fault tolerance techniques are employed during the procurement, or development, of the software. When a fault occurs, these techniques provide mechanisms to the software system to prevent system failure from occurring [18]. There are many fault tolerance techniques that deals with various kinds of faults. Re-active fault tolerance tries to reduce the effect of fault after it has occurred. There are various techniques which are based on this policy like Checkpoint/Restart, Replay and Retry and so on [32]. Checkpoint-restart techniques can be used in a manner similar to whole program restart, but can provide faster restart times since restarts are done from a checkpoint. When used in this way, these techniques still do not handle deterministic bugs, since these bugs will still occur after restarting [34]. Other uses of checkpoint-restart in conjunction with running multiple program versions have also been proposed [34] which may survive deterministic bugs if failures occur independently. However, they incur prohibitive costs for most applications in terms of developing, maintaining, and running multiple application versions at the same time.

Whereas pro-active fault tolerance tries to avoid fault at early stages before it occurs and affects the system. In cloud computing, different tasks are performed at distinct computing nodes in a virtualized environment. As the number of nodes in high-performance computing environments keeps increasing, faults are becoming common place. Reactive fault tolerance (FT) often does not scale due to massive I/O requirements and relies on manual job resubmission [9]. The principle of proactive fault tolerance policies is to avoid recovery from faults, errors and failures by predicting them and proactively replace the suspected components other working components. Techniques based on proactive FT policy are required to tolerate fault in real time computing systems running on cloud environment. Following are the techniques based on these policies:

- **Replication**

Replication can be used to create and maintain copies of an enterprise's data at these sites. When events affecting an enterprise's primary location occur, key application

services can effectively be restarted and run at the remote location—incurring no capital expenditure, only operational expenditure—until such time as the primary site is brought back online. Replication technology is available in storage arrays, network-based appliances and through host-based software [7].

- **Load Balancing**

Load balancing is often used to implement failover—the continuation of a service after the failure of one or more of its components. The components are monitored continually and when one becomes non-responsive, the load balancer is informed and no longer sends traffic to it. Due to uneven job arrival patterns and unequal computing capabilities, some nodes may be overloaded while others may be under-utilized. Load balancing mechanism aims to equally spread the load on each node of the cloud, optimizing their utilization, throughput and response. To achieve this goal, the load balancing algorithms should minimize the different between the heavily loaded and lightly loaded nodes [25].

- **Adaptive fault tolerance**

This scheme tolerates the faults on the basis of reliability of each computing node, i.e. virtual machine. A virtual machine is selected for computation on the basis of its reliability and can be removed, if does not perform well for real time applications [26].

- **Proactive Fault Tolerance using Pre-emptive Migration**

Pre-emptive Migration relies on a feedback-loop control mechanism. Application health is constantly monitored and analyzed. It is reallocated to improve its health and avoid failures. This technique is a closed- loop control similar to dynamic load balancing. There is no 100% coverage of failures that can be anticipated, such as random bit flips [28].

- **Proactive Recovery**

A Byzantine-faulty replica may appear to behave properly even when broken; therefore recovery must be proactive to prevent an attacker from compromising the service by corrupting 1/3 of the replicas without being detected. This algorithm recovers replicas periodically independent of any failure detection mechanism. However a recovering replica may not be faulty and recovery must not cause it to become faulty, since

otherwise the number of faulty replicas could exceed the bound required to provide safety. In fact, there is a need to allow the replica to continue participating in the request processing protocol while it is recovering, since this is sometimes required for it to complete the recovery [24].

- **Virtual machine migration**

Virtual Machine migration provides major benefit in cloud computing through load balance across data centers. It also provides robust and high response in data centers. Virtual machine migration was extended from process migration. Avoiding hotspots is major benefit of VM migration even though it is not straight forward. Initiating a migration lacks the facility to respond to unexpected workload changes and detecting workload hotspot.

i. **Post copy live migration**

Post-copy technique first transmits CPU's state from source to destination, and then resumes the VM at the destination. Following this step, it would combine pull and push strategies—at the target VM, pulling non existing memory pages from the source; at the source physical machine, actively pushing the VM's memory pages [35].

ii. **Pre copy live migration**

In pre-copy live migration, those frequently updated pages in iteration are marked. These pages would be sent to destination at last iteration of precopy process [36]. An improved pre-copy approach by Hierarchical Copy Algorithm has been proposed which intercepts the memory access operation of VM, records the times of write interrupt of dirty pages, and then marks them as high dirty pages if their write interrupt times exceed a preset threshold. Finally these high dirty pages would be sent to destination at last iteration to avoid repeatedly sending dirty pages [37].

2.6 Related Work on Fault Tolerance

Chen et al. [20] presented SHelp, a lightweight runtime system that can survive software failures in the framework of virtual machines. It applies weighted rescue points and error virtualization techniques to effectively make applications by- pass the faulty path. A two-

level storage hierarchy is adopted in the rescue point database for applications running on different virtual machines to share error handling information to reduce the redundancy and to more effectively and quickly recover from future faults caused by the same bugs.

Sidiroglou et al. [33] presented ASSURE, a system that introduces rescue points that recover software from unknown faults while maintaining both system integrity and availability, by mimicking system behavior under known error conditions. ASSURE restores execution to an appropriate rescue point and induces the program to recover execution by virtualizing the program's existing error-handling facilities.

Kaushal et al. [24] proposed a FT solution in the cloud at the customer level by replicating servers queries, based on the HA-Proxy load distributor. Other researches such as [10] and [11] proposed a collaborative solution for specifics applications (MPI for example). However, they do not consider the splitting of the cloud between a (VM) provider and its customers, so their works are only applicable to an SaaS cloud.

Uesheng Tan et al. [29] described a replication solution for VM FT, exclusively managed by the cloud provider. It proposes to improve efficiency by using passive VM replicas (with very few resources), which become active when a failure is detected. A mechanism is introduced to transfer/initialize the state of VM.

Wenbing Zhao et al. [13] proposed a FT middleware, which can be used by a cloud customer to implement software FT. Their main purpose is to implement a synchronized server replication strategy so that a failed server can be repaired with a consistent state.

OpenNebula offers exclusive VM FT implemented at the cloud level. It allows the cloud provider to associate hooks (scripts or programs) with each type of VM failure [21]. However hardware failures are not addressed by OpenNebula for two reasons: it provides no hardware sensors and all VM sensors are located on the same machine than the VM. So a machine failure cannot be detected by OpenNebula.

As OpenNebula, the Microsoft Windows Azure platform [22] offered an exclusive FT management at the cloud level. Windows Azure replicates each VM so that a VM failure is covered by its replicas. The Azure solution is limited to web applications developed in

the Windows Azure platform. Moreover, no solution is proposed to repair the failed VM. In addition, for VMs which are not instantiated by the Azure development platform, the entire responsibility of FT management is left to the customer. This is also the case in the Amazon EC2 [23] cloud platform.

After studying various research proposals in the area of fault tolerance in cloud environment, it was realized that fault tolerance and availability are one of the critical issues. In real time application running on cloud environment, failures may cause interruption in operations. So continues monitoring of services and prediction of fault helps to improve the reliability and availability of the system. The prediction and monitoring mechanisms has been identified to be one of the best solutions to improve the fault tolerance of the system and availability metrics in cloud environment.

Chapter 3
Problem Statement

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

3.1 Gaps Analysis

- Existing Cloud service architectures and Fault tolerance models in cloud environment are still not that capable to handle large applications.
- As per as the research gaps analyzed there is a potential need for implementing autonomic fault tolerance by using different parameters in cloud environment.
- Fault Prediction and Monitoring framework needs to be developed for real time applications that run in cloud environment.
- Issues related to data management like data consistency, data integrity, data loss etc should be handled carefully for better availability, lower latency, and other benefits.

3.2 Problem Definition

Since cloud computing has been an emerging practice and has shown a great growth in its applications and publicity, only performance and energy trade-off have been considered so far with a lower emphasis on the system dependability/availability. As per the research analysis this has been demonstrated to be the weakest link in the chain for early cloud providers but now with real time applications running in cloud environment, these attributes cannot be neglected. Therefore, an autonomic fault tolerance and prediction framework should be implemented to fill this research gap devising a backup server allocation during run time in cloud virtualized environment. The system will be capable to predict and handle faults. Another problem in cloud environment exits is the increasing criticality of data. Replication is an interesting alternative that addresses both of these issues. Data replication is highly adaptive to failures of any kind so as to maintain high data availability. A potential solution could be to maintain eventual data consistency among replicas.

The major focus of this thesis is to provide a design for fault prediction and monitoring framework which resolves the issue of availability, data replication and fault prediction.

3.2 Methodologies

- To handle fault at the server end, HAProxy has been used. It redirects the request to another server if one server fails and also balances load between various servers.
- Data replication has been achieved by using Hadoop. It uses multiple DataNodes to store data in replicated environment.
- Nagios is used to develop a fault prediction mechanism. It makes use of plug-ins to monitor Hadoop and provide prior intimation about the failure that could occur.

Chapter 4

Solution to Problem

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

4.1 Design of Solution

Following section gives overall guidance on the implementation of fault tolerant framework using various tools in cloud environment. After developing the framework, a small prototype based on this framework is developed. Implementation includes a Virtual Machine server(VMware fusion) and Linux operating system (Ubuntu 12.04) that is evaluated with two virtual machines (VMs) hosting as web servers, named as server_1 and server_2, both running the same application instances. Apache web server is configured on server 1 and server 2. A Web engine developed in Java handle data passed from web servers. HAProxy software version 1.4.24 and hadoop version 1.0.4 is configured on third virtual machine.

Data is kept in hadoop clusters in replicated environment. Two DataNodes are created per cluster. Nagios is configured on one of the virtual machines to check status of DataNodes.

4.2 Framework Design

This section presents the Framework of the fault tolerant system for web based applications. The proposed framework for the fault tolerant web based system is shown in Figure 4.1. This framework provides a better Quality of service and fault tolerance for the web applications running in cloud environment. With the combination of load balancer, multiple data clusters and by monitoring these clusters, higher availability, better Quality of service and fault tolerance is ensured. Various components of the framework are explained below:

- **User Application layer:** User can access services located in cloud environment using this layer. From this layer user's requests are received by load balancer which forwards request to the appropriate web server.

- **Load Balancer:** Load Balancer receives incoming requests for the web application and forwards those requests to the specific web server. When a web server is down or serving some other request then the Load Balancer forward this request to other server which could handle it.

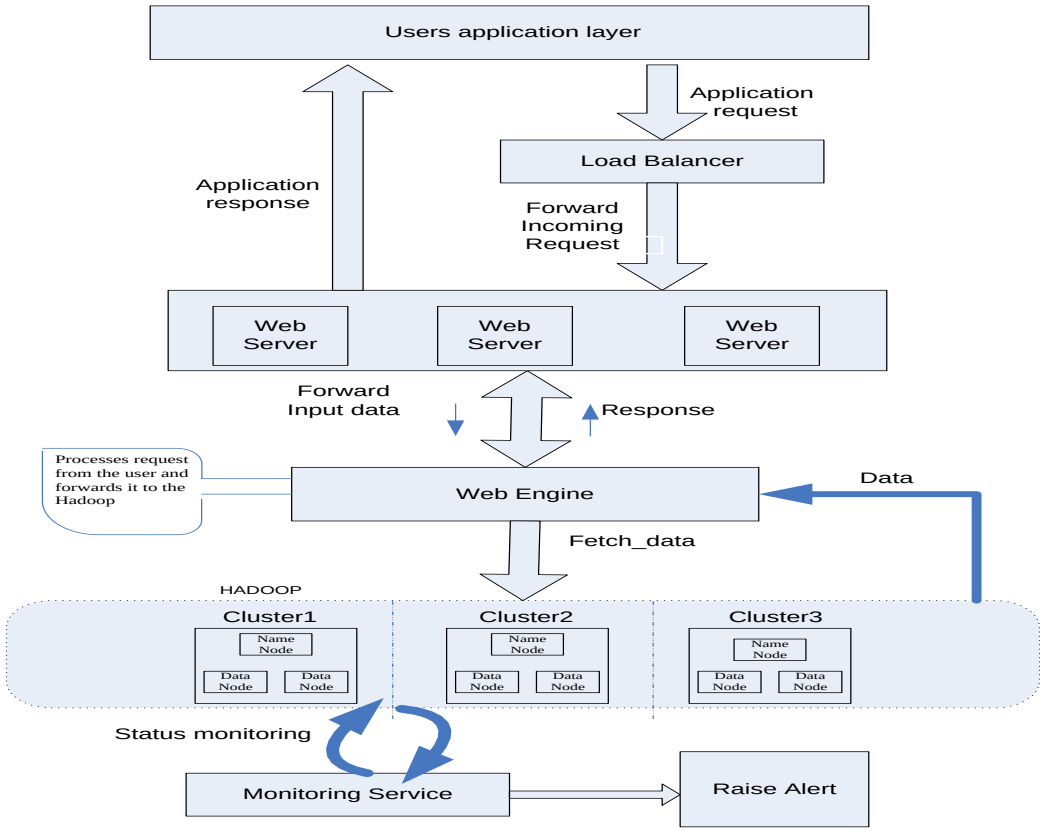


Figure 4.1: Proposed Framework

- **Web Engine:** Web engine is developed in java. It handles all the requests coming from various interfaces and provides the appropriate service needed to fulfill request. After processing request it forwards data to the storage cluster. Also when request for data arrives, it fetches data from hadoop and sends it to the user.

- **Hadoop Clusters:** Processing of data is done at various locations and at times data moves from one location to another. Because of this maintaining data becomes one of the critical tasks. These clusters are used to store data that comes from the web Engine. Clusters maintain multiple copies of data. This increases the availability. Also when required, data is fetched from the nearest DataNode which ensures higher performance. Location of data becomes an important factor when we are dealing with distributed computing environment. In case, if one node goes down then data can be easily retrieved from another node.

- **Monitor:** Monitoring tool sends alerts when critical infrastructure components fail and recover, providing administrators with notice of important events. It collects information regarding the number of clusters and DataNodes per cluster. If a DataNode or a cluster fails then it raises alert and asks for the appropriate action. Alerts can be delivered via email to a local or a remote machine.

4.3 Interaction Diagram

This section presents the sequence diagram. Figure 4.2 shows various interactions between various entities of the framework.

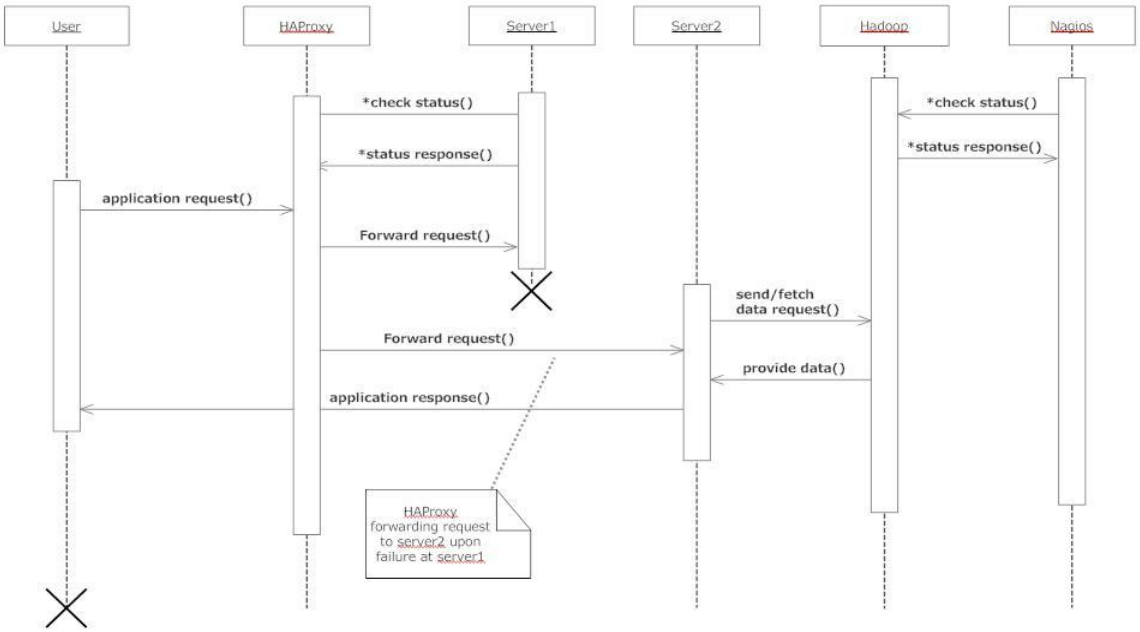


Figure 4.2: Sequence diagram

User sends application_request() at HAProxy server. HAProxy sends check status() signal to servers being monitored by HAProxy. These servers sends status_response() back to the HAProxy. This communication remain continue with servers until they fail. If any server fails, the request is directed to second server i.e server2. These servers request Hadoop cluster for data. Nagios checks DataNode's status and Hadoop responds to it. This communication goes on till DataNodes are live.

4.4 Implementation of the Framework

Following tools have been used for implementing the above proposed framework.

4.4.1 Hadoop

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 [17].

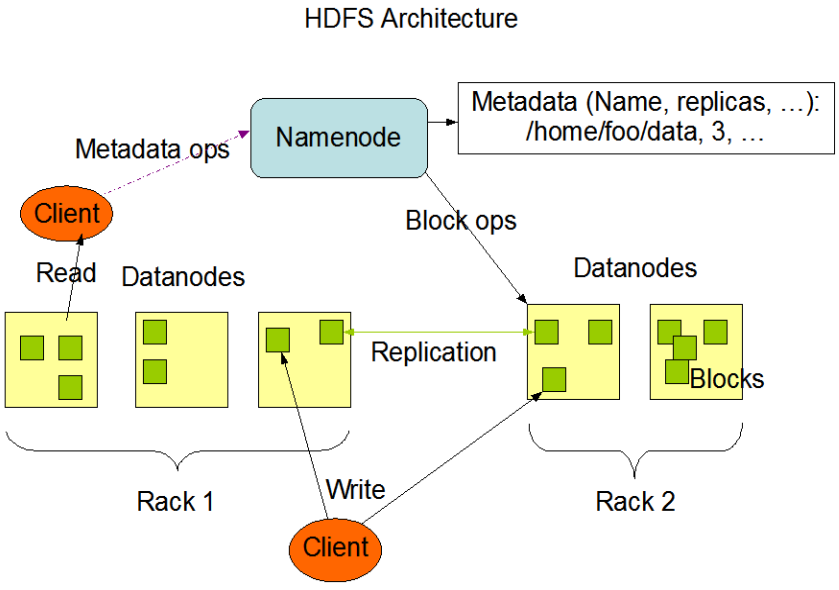


Figure 4.3 HDFS Architecture [17].

- **Hadoop Distributed File System**

The Hadoop Distributed File System (HDFS) provides global access to files in the cluster [17]. For maximum portability, HDFS is implemented as a user-level filesystem in Java which exploits the native filesystem on each node, such as ext3 or NTFS, to store data. Files in HDFS are divided into large blocks, typically 64MB, and each block is stored as a separate file in the local filesystem.

HDFS is implemented by two services: the NameNode and DataNode. The NameNode is responsible for maintaining the HDFS directory tree, and is a centralized service in the cluster operating on a single node [17]. Clients contact the NameNode in order to perform common filesystem operations, such as open, close, rename, and delete. The NameNode does not store HDFS data itself, but rather maintains a mapping between HDFS file name, a list of blocks in the file, and the DataNode(s) on which those blocks are stored.

- **MapReduce Engine**

In the MapReduce model, computation is divided into a *map* function and a *reduce* function. The map function takes a key/value pair and produces one or more intermediate key/value pairs. The reduce function then takes these intermediate key/value pairs and merges all values corresponding to a single key [17]. The map function can run independently on each key/value pair, exposing enormous amounts of parallelism. Similarly, the reduce function can run independently on each intermediate key, also exposing significant parallelism.

In Hadoop, a centralized JobTracker service is responsible for splitting the input data into pieces for processing by independent map and reduces tasks, scheduling each task on a cluster node for execution, and recovering from failures by re-running tasks. On each node, a TaskTracker service runs MapReduce tasks and periodically contacts the JobTracker to report task completions and request new tasks. By default, when a new task is received, a new JVM instance will be spawned to execute it [17].

4.4.2 HAProxy

HAProxy stands for High Availability Proxy and is used by companies such as RightScale for load balancing and server fail over in the cloud. 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 [19]. 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, failover and load balancing mechanism, and proxying 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 [19].

HAProxy is installed on all "Front End" Server Templates. A "Front End" (FE) is a server with a static IP (typically a 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 be 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 [19]. HAProxy can be easily migrated to another server in the presence of system failures.

HAProxy currently does not support the HTTP keep-alive mode, but knows how to transform it to the 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. If keep-alive or pipelining are required, HAProxy will still support them, but will only see the first request and the first response of each transaction. While this is generally problematic with regards to logs, content switching or filtering, it most often causes no problem for persistence with cookie insertion.

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.

HAProxy Configuration

HAProxy's 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 proxies sections which can take form of "defaults", "listen", "frontend" and "backend".

First parameter is an optional list of arguments which may be needed by some algorithms. Here, the load balancing algorithm of a backend is set to roundrobin. The algorithm may only be set once for each backend. 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 ‘ulimit-n’ are used for process management and security and for later ‘maxconn’ are used. Then for third parameter, "defaults" section sets default parameters for all other sections following its declaration. Those default parameters are reset by the next "defaults" section. 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 with its frontend and backend parts combined in one section. It is

generally useful for TCP-only traffic [19]. Following configurations are done in haproxy.cfg file.

#haproxy.cfg

```
defaults
maxconn         2000
contimeout      5000
clitimeout      50000
srvtimeout      50000

listen          stats :1936
mode            http
stats           enable
stats           hide-version
stats           realm Haproxy\ Statistics
stats           auth dhananjya:binny

listen          hadoop 192.168.85.201:8445
mode            http
balance         roundrobin
maxconn         10000
server          cluster1 192.168.85.200:50070 maxconn 5000 check
server          cluster2 192.168.85.202:50070 maxconn 5000 check

listen          web :2020
mode            http
balance         roundrobin
maxconn         10000
server          server_1 192.168.85.201:80 maxconn 5000 check
server          server_2 192.168.85.203:80 maxconn 5000 check
```

4.4.3 Nagios

Nagios monitors entire IT infrastructure to ensure systems, applications, services, and business processes are functioning properly. In the event of a failure, Nagios can alert technical person of the problem, allowing them to begin remediation processes before outages affect business processes, end-users, or customers [45]. Nagios has a modular design with a web interface and a set of plugins to check the different services. Nagios is not specifically an IDS. Nagios possesses a friendly interface, is easy to use, very flexible

and it is endowed a system of sending alerts. Following is the service definition and the command definition for monitoring DataNodes.

```
define command{
command_name check_remote_datanode
command_line /etc/nagios3/libexe/check_hadoop_datanode.pl $HOSTADDRESS$  $ARG1$ $ARG2$
                }

define service{
use                             generic-service         ; Name of service template to use
host_name                       localhost
service_description             check_remote_datanode
check_command                   check_remote_datanode!1!0
            }
```

Figure 4.4: Nagios service & command definition

4.4.4 VMware Fusion

With VMware Fusion™, you can run personal computer (PC) applications and devices on your Intel-based Mac. VMware Fusion takes advantage of the security, flexibility, and portability of virtual machines to run Windows and other x86 operating systems at the same time as Mac OS X.

Product Benefits:

- Run Windows and Linux applications on a Mac. You can run your favorite applications at the same time as Mac applications using virtual machines running Windows and Linux operating systems, without rebooting your Mac.
- The New Virtual Machine Assistant guides you through the process of creating a virtual machine, including Windows Easy Install and Linux Easy Install.6 Getting Started with VMware Fusion VMware, Inc.
- The integrated Migration Assistant helps you convert your physical PC to a virtual machine to run on your Mac.
- You can import virtual machines created with Parallels Desktop or Microsoft Virtual PC for Mac directly to VMware Fusion.

- VMware Fusion can use an existing Boot Camp partition, or, you can import your Boot Camp partition into a virtual disk, letting you reclaim your Boot Camp space. VMware Fusion eliminates the need to reboot to access your Windows applications and files.
- You can take multiple snapshots—pictures in time—of your virtual machines, keeping them safe in case of any problem. VMware Fusion AutoProtect takes automatic, periodic snapshots to keep your virtual machines safe from unexpected harm.

4.4.5 Ubuntu

Ubuntu is an operating system based on the Debian GNU/Linux distribution and distributed as free and open source software. It is named after the Southern African philosophy of Ubuntu ("humanity towards others"). Ubuntu is designed primarily for desktop use although net book and server editions exist as well. Ubuntu software centre gives instant access to thousands of applications that are needed to customize the computer.

Figure 4.3: Ubuntu in Virtual Environment

In this thesis, Ubuntu 12.04 LTS has been installed on all the two virtual machines. The ISO image of Ubuntu-12.04-desktop-i386 is downloaded and installed on the VMware fusion. It is fast, secure, easy to install and easy to use operating system.

4.4.6 Eclipse

Eclipse as an integrated development environment (IDE) for Java. Eclipse is created by an open source community and is used in several different areas, e.g. as IDE for Java or for Android or as a platform to develop Eclipse RCP applications, etc. Eclipse requires an installed Java Runtime.

Chapter 5

Implementation and Experimental Results

This chapter focuses on implementation of a prototype based on Fault Prediction and Monitoring Framework for applications running in cloud environment presented in previous chapter. Implementation has been done using various tools described in previous chapter. These tools are meant to tolerate various kinds of faults that could halt normal working of the complete system. This chapter shows the experimental results of this prototype.

5.1 Implementation and Working of the system

Fault tolerant system has been developed using HAProxy as a Load balancer, Hadoop as data storage cluster and Nagios as a monitoring tool. HAProxy is used to handle server failures in Fault Tolerant Cloud Environment. HAProxy provides a web interface for statistics known as HAProxy statistics. HAProxy forwards the web application requests to the intended servers. It balances load between the servers. Also if one server goes down, it redirects request to other server. Two web servers are configured to handle user requests and their status is monitored by HAProxy. Basic functionality of the system is shown as snapshots below. Figure 5.1-5.3 shows a small web application developed in HTML and running on two servers.

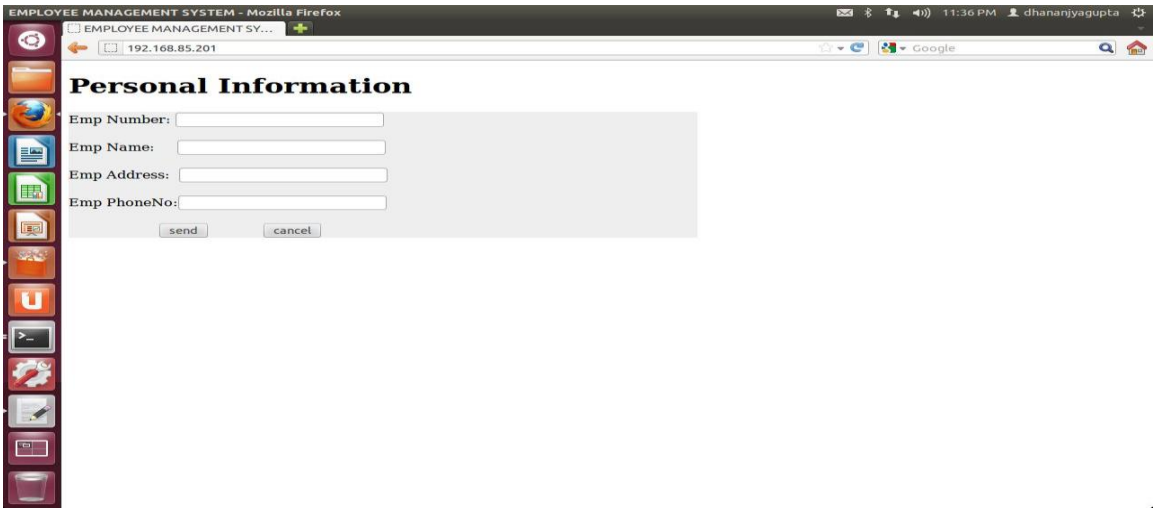


Figure 5.1: Web application page (1)

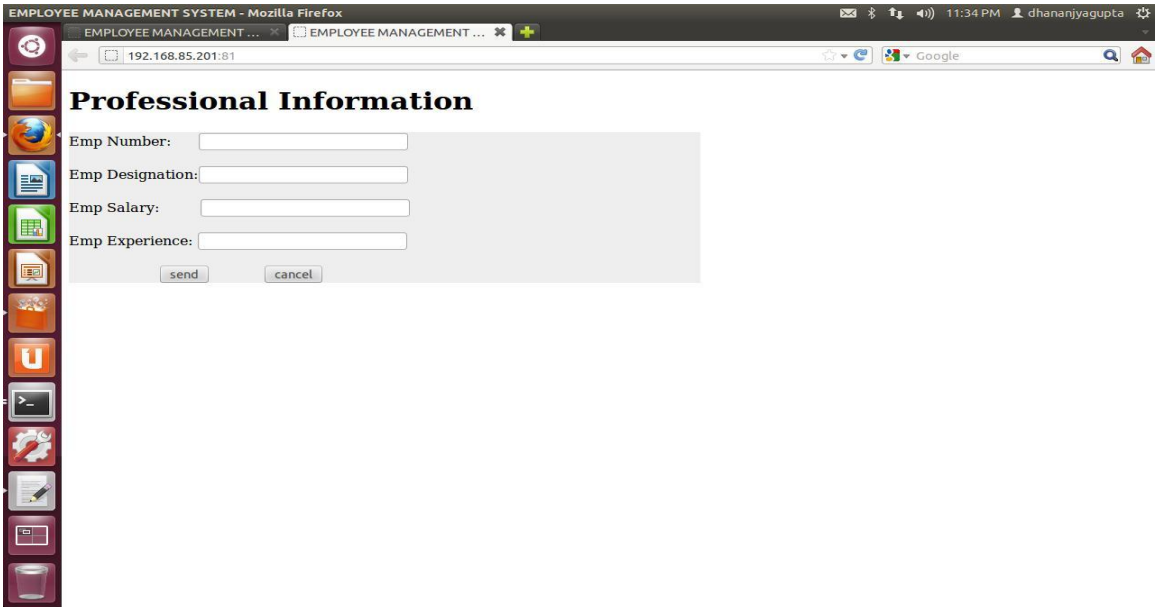


Figure 5.2: Web application page (2)

User enters data in the respective fields on both web pages and the clicks on send button. After this, data is sent to web engine which stores this data on to the hadoop clusters. This data is stored on two clusters in two replicas.

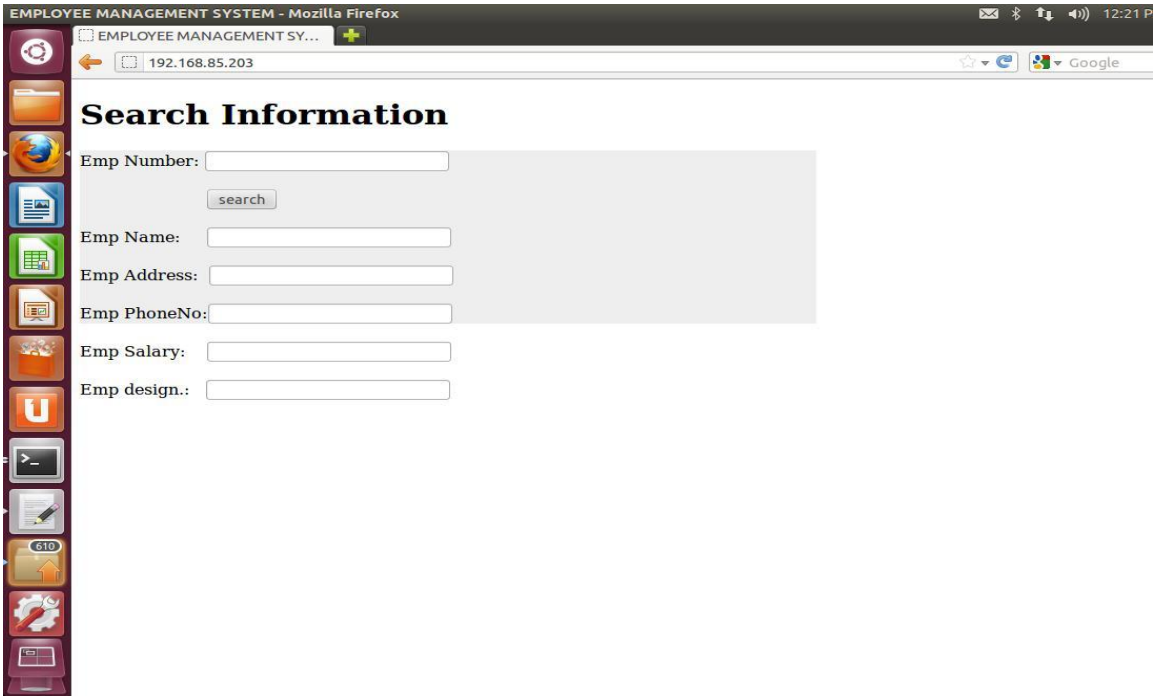


Figure 5.3 Web application page (3)

Figure 5.3 shows the web page for searching information about an employee. User enters the employee ID and then clicks search. This search request goes to web engine which runs a MapReduce job to search information of employee stored in HDFS.

Running MapReduce job

Following is the command triggered when user search for details of a particular employee. Employee ID is passed as the third parameter to the command mentioned below.

```
root@ubuntu:~/hadoop/hadoop-1.0.4/bin# ./hadoop jar FetchData-1.0.4_new.jar FetchData
../logs/input/project_data  ../webapps/output55/  801131004
```

MapReduce job is written and its jar is created named FetchData-1.0.4_new.jar. It takes three parameters. First is the input directory where input files are stored. Second is the output directory where output of MapReduce job would be written and third parameter is the ID of the employee whose record is to be searched. Figure 5.4 shows the running of MapReduce job.

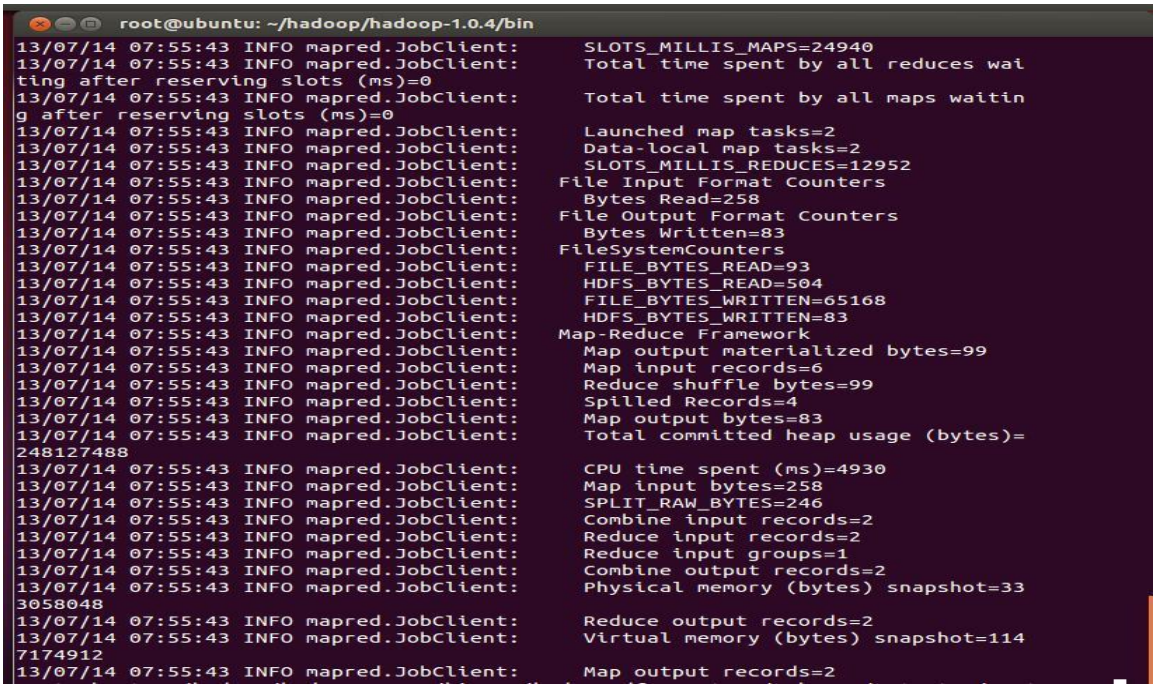


Figure5.4: Execution of MapReduce job

After MapReduce job execution completes, it stores output in a file. Figure 5.5 shows the content of output file named part-00000. File contains two records fetched from each file supplied (input files). This data is received by the web engine where it is parsed and presented to the requesting user.

```
root@ubuntu:~/hadoop/hadoop-1.0.4/bin# ./hadoop dfs -cat ../webapps/output55/par
t-00000
Warning: $HADOOP_HOME is deprecated.

801131004       801131004 amritpal #1191_sst_nagar 9876546323
801131004       801131004 System_engineer 45000 2_years
root@ubuntu:~/hadoop/hadoop-1.0.4/bin#
```

Figure 5.5: Reducer output file

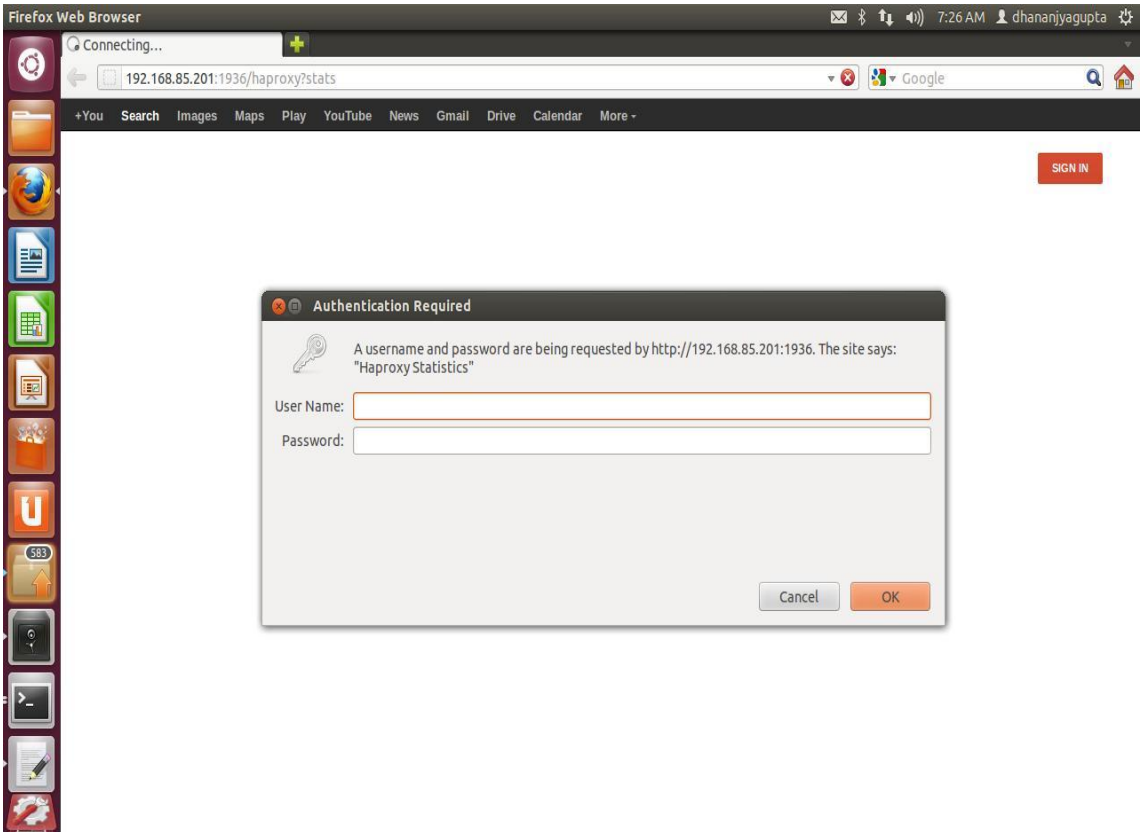


Figure 5.6: HAProxy Authentication

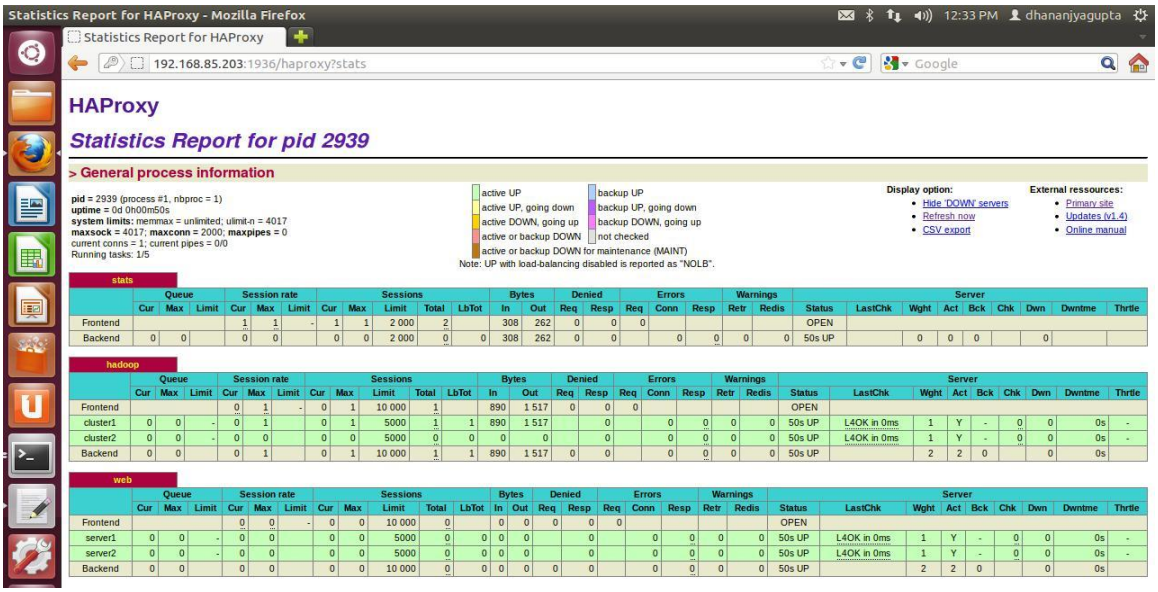


Figure 5.7: HAProxy statistics

Figure 5.6 shows HAProxy authentication window. HAProxy statistics in figure 5.7 shows that two storage clusters and two web servers are configured and their status shows that all are working properly.

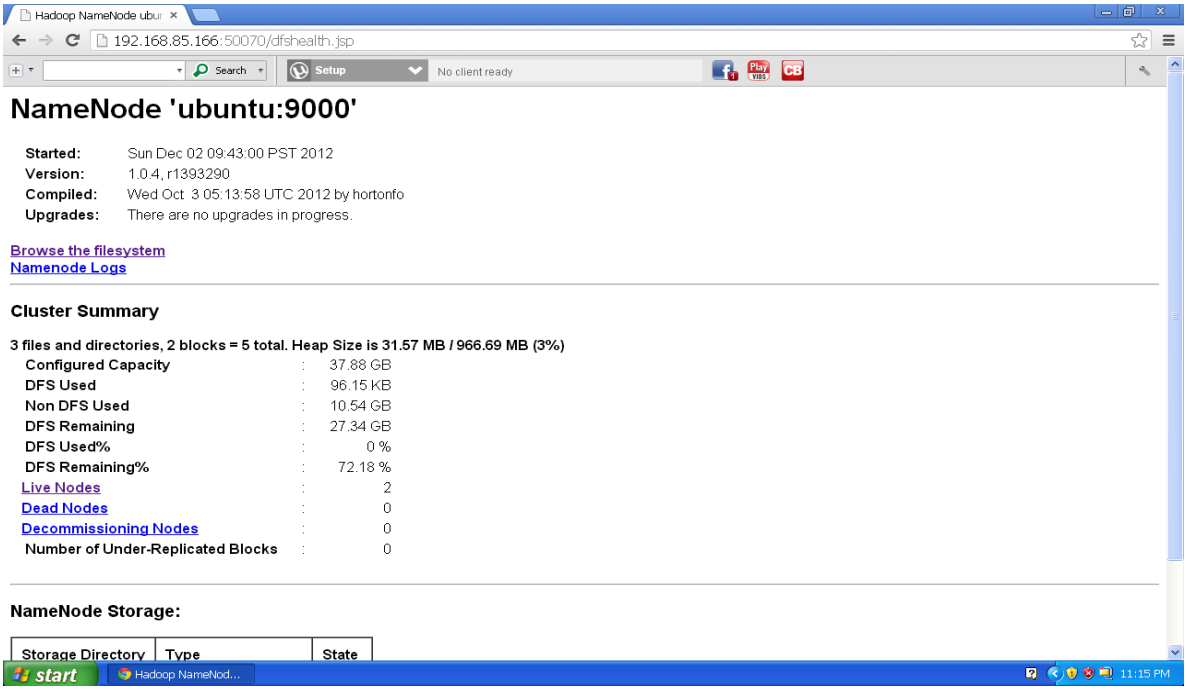


Figure 5.8: Hadoop cluster

Web interface for NameNode of one storage cluster is shown in Figure 5.8. This interface updates the information after few seconds. Under this NameNode, two DataNodes are configured and both are live. Replication of data is achieved by maintaining two copied of data on these Data Nodes. If one DataNode fails then second DataNode provides data. Figure 5.9 shows DataNodes status being monitored by Nagios. Figure 5.9 shows that both the DataNodes are up and running properly. Figure 5.10 shows various alerts raised by Nagios. If DataNode's count reaches a threshold value then Nagios changes its status to warning and if it decreases than threshold then it changes its status to critical. This status information is sent to the concerned authority via E-mail or SMS.

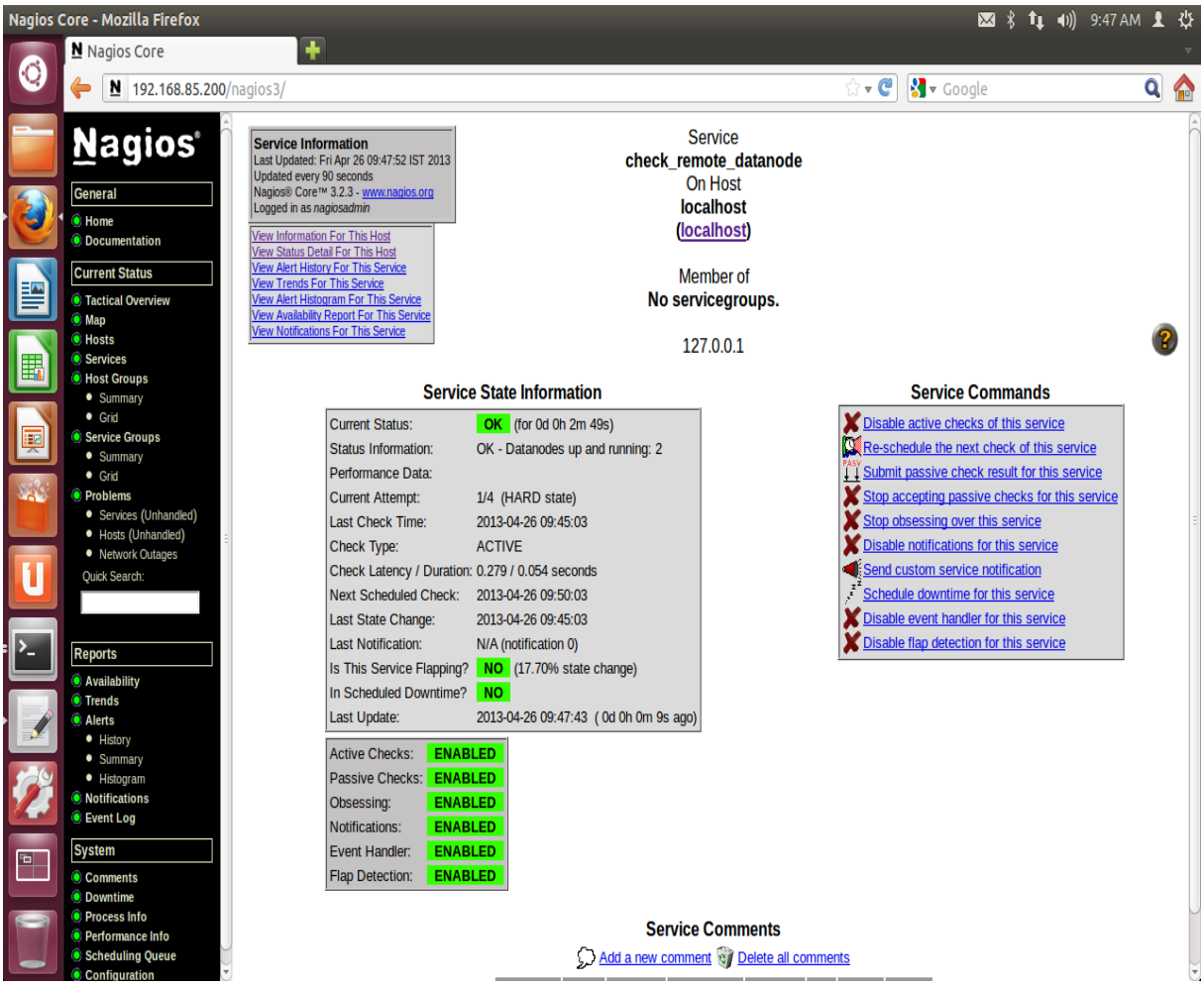


Figure 5.9: DataNodes staus by Nagios

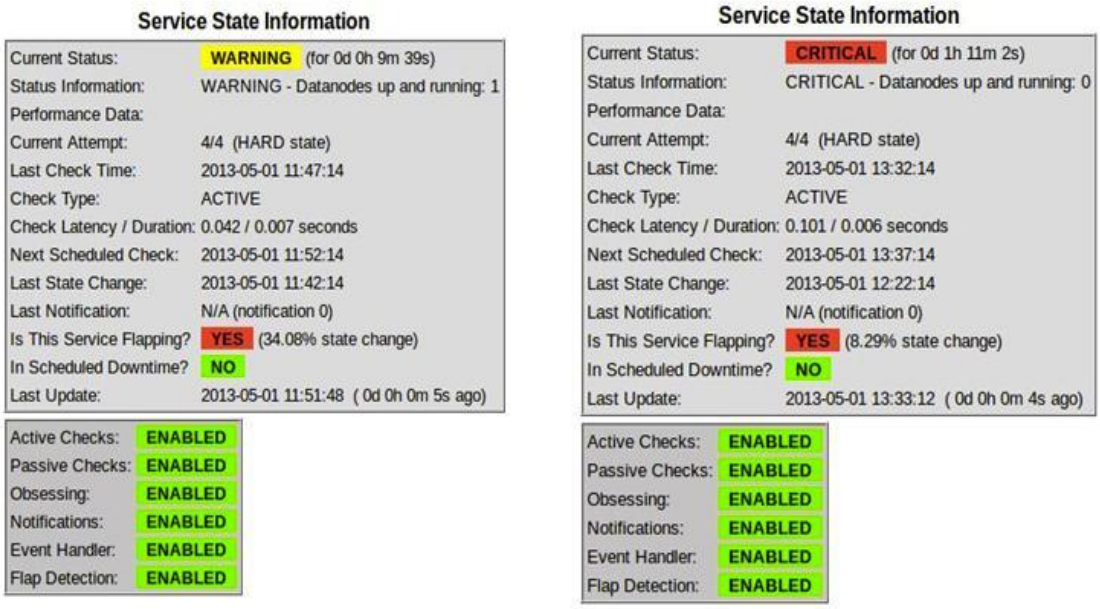


Figure 5.10: Alerts raised by Nagios

Availability Analysis

To analyze the availability of system, three DataNodes are created. Following data taken from the log file of NameNode shows the time at which all DataNodes are added, their IP addresses and the respective port numbers.

07:14:47,225 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default rack/192.168.85.205:57418

07:22:34,275 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default rack/192.168.85.206:50010

07:34:14,275 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default rack/192.168.85.207:37689

A MapReduce job is executed with large input files and during execution two nodes got failed. But following snapshots of the MapReduce job shows that job execution completed successfully and provided the correct results using single DataNode only.

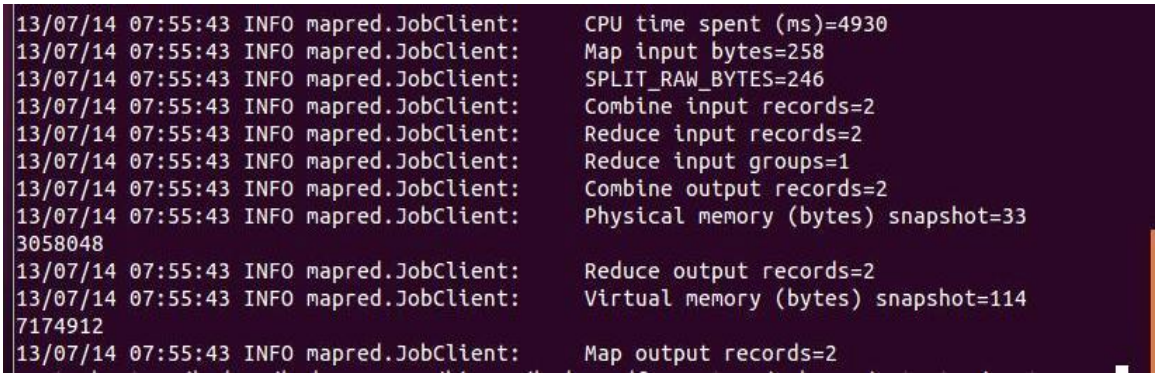


Figure 5.11: MapReduce job

Figure 5.11 shows that even after failure of two DataNodes, system is able to complete MapReduce job using single DataNode correctly. Hence it ensures the availability of the system.

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 Conclusion

In this thesis a Fault tolerant Framework has been proposed. This framework makes use of load balancing and data replication to tolerate various software faults. This framework provides fault prediction mechanism to predict faults in Hadoop cluster. Thus various faults in Hadoop cluster can be avoided by adding more DataNodes on requirement. A prototype based on this framework is deployed on multiple virtual machines. Map, Reduce methods have been implemented for manipulation of data stored in HDFS. HDFS replicates data on multiple DataNodes. The proposed framework has been evaluated using prototype having two web servers, three DataNodes in Hadoop cluster. The experimental results demonstrate that the availability of DataNodes after the failure occurs.

6.2 Thesis Contribution

- A fault prediction and monitoring framework is proposed and implemented using various tools.
- Hadoop is setup for Data replication, HAProxy is configured to handle faults occurs at web servers and Nagios is setup to predict failure in DataNodes.
- A prototype is designed, implemented and analysed using the proposed framework.

6.3 Future Research

This research work shows a limited number of DataNodes in Hadoop cluster but in real time computing, it can cover thousand of nodes. This work concentrates on availability and fault tolerance issues but it can be further extended for more critical metrics. Secondly, this work shows implementation of the single application on both the servers. Multiple applications can be included for better analysis of the framework. Real time data can be analyzed and monitored using tools presented in this work.

References

[1] David C. Wyld, “Moving to the cloud: An Introduction to Cloud Computing in Government”, IBM centre for The Business of Government e-Goverment Series, 2009.

[2] Peter Mell, Timothy Grance “The NIST Definition of Cloud Computing”, NIST Special Publication 800-145, September, 2011.

[3] Buyya R, Ranjan R. Special Section "Federated Resource Management in Grid and Cloud Computing Systems". Future Generation Computer Systems, 26(8): pages 1189-1191,2010.

[4] Wayne P. "Top Business and Operational Advantages Cloud Computing Can Deliver to IT Organizations", SunGard Availability Services, 2010.

[5] X. Chu et al, "Aneka: Next-generation enterprise grid platform for e-science and e-business applications”, Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing, 2007.

[6] Ghalem Belalem, Said Limam, “Fault Tolerant Architecture to Cloud Computing Using Adaptive Checkpoint,” International Journal of Cloud Applications and Computing, 1(4), pp 60-69, 2011.

[7] H.Yu, A. Vahdat, "Consistent and automatic replica regeneration". Trans. Storage 1, pages 3–37, 2005.

[8] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu, "Cloud Computing and Grid Computing 360-Degree Compared”, 2008.

[9] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, " Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems", 25(6): Elsevier Science, Amsterdam, the Netherlands, pages 599-616, June 2009.

[10] D. Gartner, "Seven cloud-computing security risks". InfoWorld.-07-02. http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853. 2008

[11] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou, Weizhong Qiang, Gang Hu, “SHelp: Automatic Self- healing for Multiple Application Instances in a Virtual Machine Environment”, IEEE International Conference on Cluster Computing, 2010.

[12] Qi Zhang, Lu Cheng, Raouf Boutaba, “Cloud computing: state-of-the-art and research challenges,” Springer, The Brazilian Computer Society, vol 1, pages 7-18, 2010.

[13] J. E. Smith and R. Nair, “Virtual Machines: Versatile platforms for systems and processes”, Morgan Kauffmann, 2005.

[14] R. Buyya and M. Murshed, "GridSim: 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.

[15] "Cloud Simulation Frameworks" [online] available: http://cloud-simulation-frameworks.wikispaces.asu.edu [Accessed 26 March 2013]

[16] Buyya R., “Market-Oriented Cloud Computing: Vision, Hype and Reality for Delivering IT Services as Computing Utilities” 2008.

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

[18] Zaipeng Xie, Hongyu Sun and Kewal Saluja, “A Survey of Software Fault Tolerance Techniques”. 2004

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

[20] “Cloud Deployment Model,” [online] Available: http://www.google.com, [Accessed: 15 February, 2012].

[21] OpenNebula, “Opennebula.org: The open source toolkit for cloud computing,” [online] Available: http://opennebula.org, [Accessed: 16 january, 2013].

[22] Microsoft, “Windows azure: Microsoft’s cloud services platform,” [online] Available: http://www.microsoft.com/windowsazure, [Accessed: 16 january, 2013].

[23] "Amazon, Inc, Amazon Elastic Compute Cloud (Amazon EC2)", [online] Available: http://aws.amazon.com/ec2/#pricing, [Accessed: 21 january, 2013].

[24] Vishonika Kaushal, Anju Bala, “Autonomic fault tolerance using haproxy in cloud environment,” International Journal of Advanced Engineering Sciences and Technologies, vol. 7, 2010.

[25] Jasma Balasangameshwara, NedunchezhianRaju, “A hybrid policy for fault tolerant load balancing in grid computing environments” Journal of Network and Computer Applications,2011

[26] Sheheryar Malik, Fabrice Huet, “Adaptive Fault Tolerance in Real Time Cloud Computing”, IEEE World Congress on Services,2011.

[27] Imad M. Abbadi, “Self-Managed Services Conceptual Model in Trustworthy Clouds' Infrastructure”, 2010.

[28] Antonina Litvinova, Christian Engelmann and Stephen L. Scott,” A Proactive Fault Tolerance Framework For High Performance Computing”, 2009.

[29] Uesheng Tan, Dengliang Luo, and Jingyu Wang, “Cc-vit: Virtualization intrusion tolerance based on cloud computing,” in 2nd International Conference on Information Engineering and Computer Science (ICIECS), pp. 1-6. Wuhan, China, 2010.

[30] Chao Wang1, Frank Mueller, Christian Engelmann, Stephen L. Scott, “Proactive Process-Level Live Migration in HPC Environments,” New Mexico, Oct 2008.

[31] Yaakoub El Khamra , Yaakoub El Khamra ,“Developing Autonomic Distributed Scientific applications: A Case Study From History Matching Using Ensemble Kalman-Filters”, GMAC’09, June 15, 2009.

[32] Anju Bala, Inderveer Chana, “Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing”, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[33] S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, and A. D. Keromytis, “ASSURE: Automatic Software Self-healing Using REscue points”, Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS’09), ACM Press,Washington, DC, USA, pp.37-48, March 7-11, 2009.

[34] T. C. Bressoud and F. B. "Schneider. Hypervisor-Based Fault Tolerance". In Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP 1995), pages 1–11, Dec. 1995.

[35] M. R. Hines, U. Deshpande, and K. Gopalan, “Post-Copy Live Migration of Virtual Machines,” ACM SIGOPS Operating Systems Review, Volume 43 Issue 3, July 2009.

[36] F. Ma, F. Liu, and Z. Liu, "Live Virtual Machine Migration Based on Improved Pre-copy Approach," Proc. Software Engineering and Service Sciences, pp. 230-233, 2010.

[37] Z. B. Liu, W. Y. Qu, T. Yan, and H. T. Li, "Hierarchical Copy Algorithm for Xen Live Migration," Proc. The 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 361-364, 2010.

[38] Russell Craig, Jeff Frazier, Norm Jacknis, Seanan Murphy, Carolyn Purcell, Patrick Spencer, JD Stanley," Cloud Computing in the Public Sector: Public Manager's Guide to Evaluating and Adopting Cloud Computing", Cisco (IBSG), November 2009.

[39] INFOSYS, "Cloud Computing". Business Innovations through Technology, SETLabs Briefings Volume No 7, 2009.

[40] "Cloud computing issues and impacts", [online] on Global Technology Industry Discussion Series, page no. 7 [Accessed on 2 july, 2013].

[41] Jim sabogal, "Cloud Computing and Life Science IT", L&T Infotech, page no. 4, accessed on 3rd july, 2013.

[42] Kevin Hamlen, Murat Kantarcioglu, Latifur Khan, Bhavani Thuraisingham, "Security Issues for cloud computing", International Journal of Information Security and Privacy, 4(2), 39-51, April-June 2010.

[43] QingqingFeng, Jizhong Han, Yun Gao, Dan Meng"Magicube: High Reliability and Low Redundancy Storage Architecture for Cloud Computing" IEEE Seventh International Conference on Networking, Architecture, and Storage. 2012.

[44] Alain Tchana, Laurent Broto, Daniel Hagimont, "Fault Tolerant Approaches in Cloud Computing Infrastructures", The Eighth International Conference on Autonomic and Autonomous Systems, ICAS, 2012.

[45] "Nagios Manual "[online] Available: http://www.nagios.org. [Accessed 13 March 2013]

List of Publication

1. Dhananjaya gupt, Anju bala, “Autonomic Data Replication in Cloud Environment ” in the International Journal of Electronics and Computer Science Engineering (IJECSE) (ISSN NO-2277-1956), Volume 2 Number 2 (April 2013) Pages Number –459-464.

2. Dhananjaya gupt, Anju Bala, “Autonomic Fault Tolerant Framework for Web Applications” in the International Journal of Computer Science and Technology (IJCST), Volume 4 Issue 2, June 2013.