# PROACTIVE FAULT TOLERANCE IN CLOUD USING MULTI LAYER FEED FORWARD ARTIFICIAL NEURAL NETWORK

A Dissertation Proposal Submitted

By

**Zeeshan Amin**

to

**Department Of Computer Science and Engineering**

In the partial fulfillment of the Requirement for the
Award of the Degree
of

**Master of Technology in Computer Science and Engineering**

**Under the guidance of**

**Asst. Prof. Harshpreet Singh**

**Under the co-guidance of**

**Asst. Prof Charanpreet Kaur**

**(May 2015)**

# ABSTRACT

With the immense growth of internet and its users, Cloud computing, with its incredible possibilities in ease, QOS and on-interest administrations, has turned into a guaranteeing figuring stage for both business and non-business reckoning customers. It is an adoptable technology as it provides integration of software and resources which are dynamically scalable. Cloud computing is built upon virtualization, distributed computing, utility computing, and more recently networking, web and software services

The dynamic environment of cloud results in various unexpected faults and failures. The ability of a system to react gracefully to an unexpected equipment or programming malfunction is known as fault tolerance. In order to achieve robustness and dependability in cloud computing, failure should be assessed and handled effectively. Various fault detection methods and architectural models have been proposed to increase fault tolerance ability of cloud.

The objective of this thesis is to propose a proactive fault tolerance approach using Artificial Neural Network which will overcome the gaps of previously implemented algorithms and provide a fault tolerant model.

# CERTIFICATE

This is to certify that Zeeshan Amin has completed M.Tech dissertation proposal titled **Proactive Fault Tolerance In cloud Using Artificial Neural Networks** under my guidance and supervision. To the best of my knowledge, the present work is the result of her original investigation and study. No part of the dissertation proposal has ever been submitted for any other degree or diploma.

The dissertation proposal is fit for the submission and the partial fulfillment of the conditions for the award of M.Tech Computer Science & Engg.

Date: 04-05-2014

Signature of Advisor:

Name: Harshpreet singh

UID: 17478

# ACKNOWLEDGEMENT

I would like to express my gratitude to my guide Mr. HARSHPREET SINGH Assistant Prof. of Computer Science & Engineering, LPU, Jalandhar, India for his generous guidance, continuous encouragement and estimated supervision. He has shown the attitude and substance of a genius as he has continually and persuasively conveyed a spirit of adventure in regard to research.

I would also like to thank my friend Satinderjeet Singh, who have supported me throughout my work, by keeping me harmonious and helping me putting pieces together.

Most importantly I would like to thank my Parents and Almighty for showing me the right direction, helping me to stay calm in oddest of the times and being my support always.

ZEESHAN AMIN

# DECLARATION

I hereby declare that the dissertation proposal entitled, **PROACTIVE FAULT TOLERANCE IN CLOUD USING ARTIFICIAL NEURAL NETWORK**, submitted for the Master of Technology (M. tech) degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date: ____________________ **Investigator: Zeeshan Amin**

**Registration No: 11102960**

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

# INTRODUCTION

This chapter provides an overview of Cloud Computing, Fault tolerance in Cloud and various Fault tolerance techniques. It also provides a brief introduction of Artificial Neural Networks and their architecture.

## 1.1 Introduction to Cloud Computing

Cloud computing refers to the delivery of computing resources over the Internet. Instead of keeping information on our own hard drive or upgrading applications for our needs, we can utilize a service over the Internet, at an alternate area, to store our data and Use its applications. The thought of cloud computing is focused around a basic idea of reusability of IT abilities [Foster et al, 2008].

Cloud computing is built upon virtualization, distributed computing, utility computing, and more recently networking, web and software services. Individuals and organizations use hardware and software managed by third parties at remote location. Online file storage, social networking sites, webmail, and online business applications are some common cloud services. User can use these services without knowing the underlying hardware and software details [Mell et al, 2009].

A real time system can utilize the immense computing capabilities and virtualized environment of cloud for the execution of tasks. On the other side, most of these are safety critical systems which require high reliability and high level of fault tolerance for their execution.

The term cloud computing has been characterized from numerous points of view by examiner firms, scholastics, industry experts, and IT organizations.

As per the official **National Institute of Standards and Technology [NIST] definition(Mell et al, 2009)**, "*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.*"

The NIST definition records five fundamental qualities of cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity or expansion, and measured service. It additionally lists three "service models" (software, platform and infrastructure), and four "deployment models" (private, community, public and hybrid) that together categorize ways to deliver cloud services.

**Gartner (Plummer et al, 2008)** has defined cloud computing "as *a style of computing in which massively scalable IT related capabilities are provided "as a service" using internet technologies to multiple external customers."*

According to **Merrill lynch "***cloud computing is the idea of delivering PERSONAL (e.g. email, presentations, word processing) and business productivity applications (e.g. sales force-automation, customer service, and accounting) from centralized servers."*

**Foster et al. (2008)** define cloud computing as:

"*A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted virtualized, dynamically-scalable, managed computing power, storage, platforms and services are delivered on demand to external customers over the internet".*

### 1.1.1 Cloud Components

Cloud computing is made up of several elements. Each element has a purpose which plays specific roles which can be classified as clients, Distributed servers, data centers.

- **Clients:** These are typically the computers which are used by the end users i.e. the devices which can be used by the end user to manage the information on cloud (laptops, mobile phones, PADs etc.)
- **Data center:** These are collection of servers where the service is hosted. In order to create number of virtual server on one physical server in data center, virtualization is used.
- **Distributed servers:** These are servers which are located in different geographical place. It provides better accessibility, security to the user.
- **Data center:** These are collection of servers where the service is hosted. In order to create number of virtual server on one physical server in data center, virtualization is used.

- **Distributed servers:** These are servers which are located in different geographical place. It provides better accessibility, security to the user.

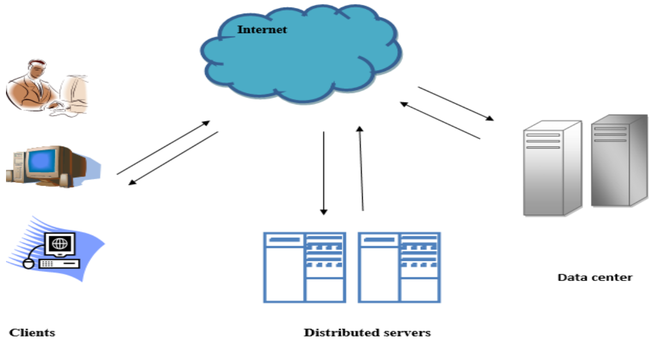


Figure 1.1: Cloud Components

### 1.1.2 Characteristics of Cloud Computing

There are ten characteristics of cloud computing in their sum up: device and location independence, scalability, on-demand services, guaranteed Quality of Service (QOS), pricing, virtualization, multi-tenancy, security and fault tolerant [Gong et al, 2010].

- **Scalability and on-demand services:** users are given on-demand resources and services over cloud. Moreover the resources provided are scalable over several data centers
- **User-centric interface:** cloud interfaces are not dependent on location of user. They can be accessed by well-established interfaces such as web services and internet browsers.
- **Guaranteed Quality of Service (QOS):** Cloud computing assures Quality of service for users by guaranteed performance, bandwidth and memory capacity.
- **Autonomous system:** users can reconfigure and combine software and information according to their requirements.
- **Cost**: No capital expenditure or any up-form investment is required in cloud. Payment for services is made on the basis of need.
- **Virtualization:** Utilization of resources is increased by sharing the server and storage devices.

- **Multi-tenancy: S**haring of resource and cost among large number of users increase efficiency and allows for
  Centralization and peek lock capacity.
- **Loose coupling:** The resources are loosely coupled as one resource functionality hardly affects the functioning of another resource.
- **Reliable Delivery**: TCP/IP is used for delivery of information between resources. Private network protocols are used within the cloud infrastructure but most of the user are connected using HTTP protocol.
- **High Security:** This is maintained on the above discussed characteristics. Loose coupling enables the jobs to execute run well, even if part of cloud is destroyed. Virtualization and abstraction of cloud provider avoid exposing the detailing of implementation.

### 1.1.3 Cloud Computing Benefits

Cloud computing reduces the response time and running time of job, also minimizes the risk in deploying application, lowered cost of deployment, and decreasing the effort and increasing innovation [Carolan et al, 2009].

- **Increased Throughput:** Cloud makes use of thousands of servers to finish an assignment in reduced time unit verses the time required by a solitary server.
- **Minimize infrastructure risk**: Cloud can be used by organizations to reduce the load of purchasing physical servers. The issues of high investment and deployment of servers depending upon the workload can be resolved considering investment on infrastructure for those application's whose attainment is short-lived.
- **Lower cost of entry:** Various characteristics outlined in previous section of cloud reduces the cost for organizations to enter new markets:
  - The capital investment is reduced to zero by renting the infrastructure instead of purchasing it and hence controls the cost.
  - The rapid application development helps to reduce the time to market, possibly giving organizations an edge against the competition.

- **Focus on innovation:** Organization relived with the issue to infrastructure deployment can focus in innovating items.

### 1.1.4 Cloud Computing Models

Services offered by the cloud providers can be grouped into three categories [Furht et al, 2010]:

- **Software as a Service (SaaS):** In this model, a complete application is provided on demand to the user. Multiple end users are serviced while at the back end a single instance of service is executed. Customers need not to go for any upfront investments, since just a single application is to be facilitated & kept up.  Google, Salesforce, Microsoft, Zoho etc are the providers of Saas.
- **Platform as a Service (Paas):** In this model, software or development environment is offered as a service. The customer is given with the option to construct his own particular applications, which run on the suppliers' base. A predefined combination of OS and application servers is provided to the user. Google's App Engine, Force.com are providing a platform to users.
- **Infrastructure as a Service (Iaas):** Standardized services that are provided are Fundamental storage and computing capabilities. Various resources are made available and shared among users in order to manage workload. The customer has to deploy his own software on the infrastructure. Amazon, GoGrid, 3 Tera are examples of Iaas.

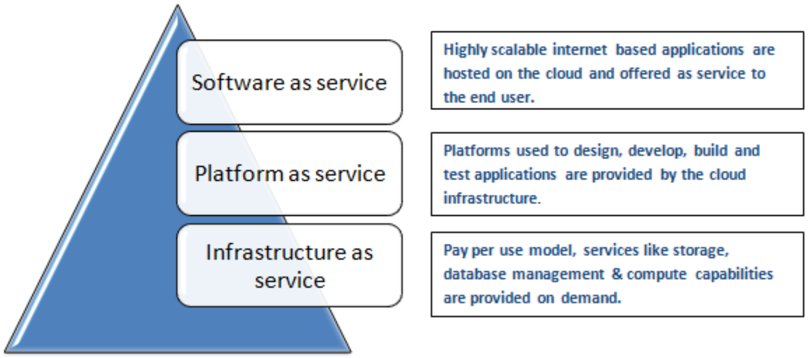

Software as service
Highly scalable internet based applications are hosted on the cloud and offered as service to the end user.
Platform as service
Platforms used to design, develop, build and test applications are provided by the cloud infrastructure.
Infrastructure as service
Pay per use model, services like storage, database management & compute capabilities are provided on demand.

Figure 1.2: Cloud Computing Models

### 1.1.5 Types of Cloud

On the bases of access to clouds, they can be classified into following types [Furht et al, 2010]:

- **Public Cloud:** Users connected to internet and having access to the cloud space can use public cloud. It refers to availability of computing resources to anyone on "Pay As You Go Basis". Public clouds are owned and operated by third parties; they deliver superior economies of scale to customers. All customers share the same infrastructure pool with limited configuration, security protections, and availability variances.
- **Private Cloud:** A private cloud in an organization is specific and limited access to a particular group. It can be referred as computing services delivered exclusively for the use of a particular organization or a group. It utilizes the same architecture for scalability and availability as the public cloud but it is limited to a single organization. Two major concerns on data security and control are addressed which are not there in public cloud.
- **Hybrid Cloud**: A combination of public and private cloud is named as hybrid cloud. With a Hybrid Cloud, service providers can expand the adaptability of computing by utilizing other Cloud Providers in full or partial manner. The Hybrid cloud environment is capable for providing on-demand, externally provisioned scale with the capacity to enlarge a private cloud to deal with any sudden surges in workload.

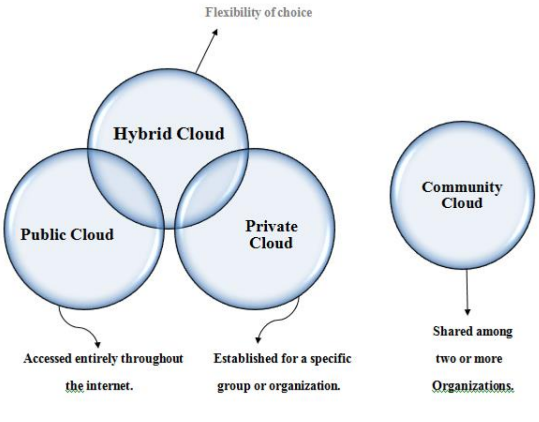

Flexibility of choice
Hybrid Cloud
Public Cloud
Private Cloud
Community Cloud
Accessed entirely throughout the internet.
Established for a specific group or organization.
Shared among two or more Organizations.

Figure 1.3: Cloud Computing Deployment Models

- **Community Cloud:** The organization with common prerequisites share the cloud functionality making it a hybrid cloud. It reduces the capital consumption by imparting the cost among the associations. The operation may be in-house or with an outsider on the premises.

## 1.2 FAULT TOLERANCE IN CLOUD COMPUTING

Fault Tolerance alludes to a methodology to system design that permits a system to keep performing actually when one of its parts falls flat or it can be defined as capacity of a system to react nimbly to an unexpected equipment or programming break down. If not fully operational, fault tolerance solutions may allow a system to continue operating at reduced capacity rather than shutting down completely following a failure [Kaushal et al, 2010].

### 1.2.1 Metrics for Fault Tolerance in Cloud Computing

The existing fault tolerance technique in cloud computing considers various parameters: throughput, response-time, scalability, performance, availability, usability, reliability, security and associated over-head [Patra et al, 2013].

- **Throughput**–It defines the number of tasks whose execution has been completed. Throughput of a system should be high.
- **Response Time-** Time taken by an algorithm to respond and its value should be made minimized.
- **Scalability**– Number of nodes in a system does not affect the fault tolerance capacity of the algorithm.
- **Performance**– This parameter checks the effectiveness of the system. Performance of the system has to be enhanced at a sensible cost e.g. by allowing acceptable delays the response time can be reduced.
- **Availability**: Availability of a system is directly proportional to its reliability. It is the possibility that an item is functioning at a given instance of time under defined circumstances.

- **Usability**: The extent to which a product can be used by a user to achieve goals with effectiveness, efficiency, and satisfaction.
- **Reliability**: This aspect aims to give correct or acceptable result within a time bounded environment.
- **Overhead Associated**: It is the overhead associated while implementing an algorithm. Overheads can be imposed because of task movements, inter process or inter-processor communication. For the efficiency of fault tolerance technique the overheads should be minimized.
- **Cost effectiveness**: Here the cost is only defined as a monitorial cost.

### 1.2.2 Fault Taxonomy

Cloud is prone to faults and they can be of different types. Various fault tolerance techniques can be used at either task level or workflow level to resolve the faults [Bala at el, 2012].

**i) Reactive fault tolerance:** Reactive fault tolerance techniques are used to reduce the impact of failures on a system when the failures have actually occurred. Techniques based on this policy are checkpoint/Restart and retry and so on.

- **Check pointing/Restart-** The failed task is restarted from the recent checkpoint rather than from the beginning. It is an efficient technique for large applications.
- **Replication:** In order to make the execution succeed, various replicas of task are run on different resources until the whole replicated task is not crashed. HAProxy, Haddop and AmazonEc2 are used for implementing replication.
- **Job migration:** On the occurrence of failure, the job is migrated to a new machine. HAProxy can be used for migrating the jobs to other machines.
- **SGuard:** It is based on rollback recovery and can be executed in HADOOP, Amazon Ec2.
- **Retry:** This task level technique is simplest among all.
  The user resubmits the task on the same cloud resource.
- **Task Resubmission:** The failed task is submitted again either to the same machine on which it was operating or to some other machine.

- **User defined exception handling:** Here the user defines the specific action of a task failure for workflows.
- **Rescue workflow**: It allows the system to keep functioning after failure of any task until it will not be able to proceed without rectifying the fault.

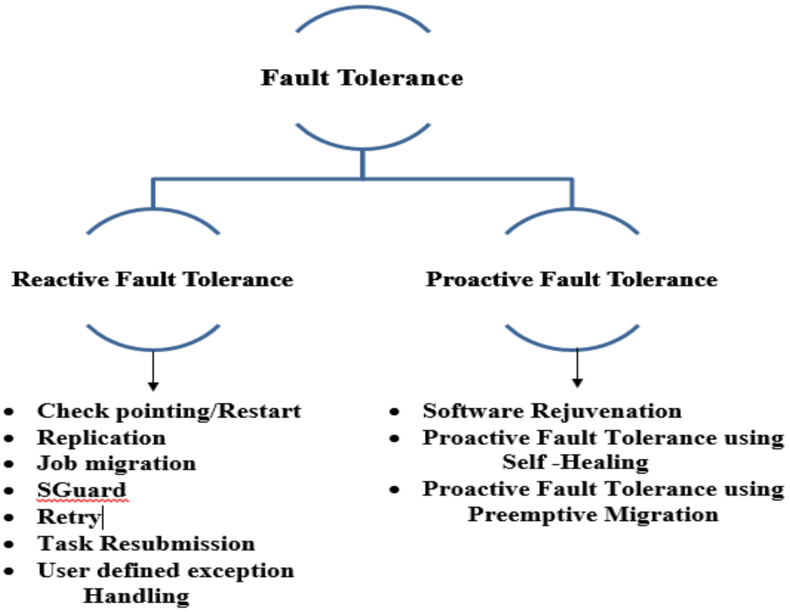

Fault Tolerance
Reactive Fault Tolerance
Proactive Fault Tolerance
Check pointing/Restart
Replication
Job migration
SGuard
Retry
Task Resubmission
User defined exception Handling
Software Rejuvenation
Proactive Fault Tolerance using Self -Healing
Proactive Fault Tolerance using Preemptive Migration

Figure 1.4: Fault Taxonomy

Due to the complex and virtual nature of cloud, reactive fault tolerance strategies are not able to manage the faults efficiently. Various problems are associated with each technique. In case of check-pointing there are two major issues that need to be solved (Dawei sun et al., 2013)

1. How often checkpoints should be inserted and save system running states.

   If checkpoints are inserted too frequently it will result in large check pointing overheads such as storage and the infrequent insertion of checkpoints will lead to more fault recovery overhead because when the failure will occur system will need to rollback with large number of operations.

2. Whether full, incremental or hybrid check pointing strategy should be applied.

   - In Full check pointing strategy whole system running state is saved to the platform periodically which makes recovery from a fault easier as the system is recovered from latest checkpoint instead from the start and thus reduces the re-computing time. However saving whole system running states results in large check pointing overheads.

- In Incremental check pointing, only first checkpoint saves complete system states then the following checkpoints only save the changes that have occurred from previous checkpoint reducing the overhead. However it will be associated will large recovery overhead as the system will have to recover form the first checkpoint.
- Hybrid check pointing combines both the strategies and has a tradeoff between check-pointing and recovery overhead. After a failure the system is recovered from the most recent full checkpoint. Thus in check pointing fault tolerance strategy a balance should be maintained between frequent and infrequent checkpoints, check pointing and fault tolerance overhead.

Replication is associated with following problems:

1. When and which data is to be replicated? Wrong selection of data and too early replication will not lead to better results.
2. How many replicas of particular data should be maintained? Large number of replicas will result in increase in system maintenance cost and it may or may not increase the availability instead lead to unnecessary spending
3. Where the replicas should be placed? System task execution can be improved by keeping all the replicas active, however in case of large scale, distributed and virtualized environments the process of replica replacement is complicated.

**Proactive Fault Tolerance:** Proactive fault tolerance predicts the faults proactively and replace the suspected components by other working components thus avoiding recovery from faults and errors. Preemptive migration, software rejuvenation etc. follow this policy.

- **Software Rejuvenation-** the system is planned for periodic reboots and every time the system starts with a new state.
- **Proactive Fault Tolerance using self-healing:**
  Failure of an instance of an application running on multiple virtual machines is controlled automatically.

- **Proactive Fault Tolerance using Preemptive Migration:** In this technique an application is constantly observed and analyzed. Preemptive migration of a task depends upon feed-back-loop control mechanism.

## 1.3 Failure Detector

A failure detector is an application or a system that is used to detect node failures or crashes. Failure detector can be classified as reliable or unreliable on the basis of result it produces. If the output of failure detector is always accurate it is called as reliable failure detector. An unreliable failure detector is one that provides information that is not necessarily accurate and it may take very long time for detection of faulty process and produce false results by suspecting the processes that have not crashed. Most of the failure detectors fall in this category.

### 1.3.1 Correctness properties of failure detectors:

- **Completeness:** when a process fails that process is eventually detected by at least one other non-faulty process. Completeness describes the capability of failure detector of suspecting every failed process permanently.
- **Accuracy:** There are no mistaken failure detections i.e. when a process is detected as failed, it has actually failed. Less number of false positives result in high accuracy. It is impossible to build a failure detector over a realistic network that is 100% accurate and complete.

  Real life failure detectors guarantee 100% completeness but the accuracy is either partial or probabilistic. There is a trade-off between completeness and accuracy
- **Speed:** Time for the detection of failure should be as less as possible. In other words, time between occurrence of a failure and its prediction must be small.
- **Scale:** There should be low and equally distributed load on each process in a group and also low overall network load.

  A failure detector should guarantee all of these properties in spite of the fact that there can be arbitrary simultaneous multiple process failures. In addition to these properties Chandra and Toug (Chen et al, 2002) proposed a set of metrics that specify Quality of service (QOS) of a failure detector.

- **Detection time (TD):** Time that elapses from crashing of a process p to the time when another process q starts suspecting process p permanently.

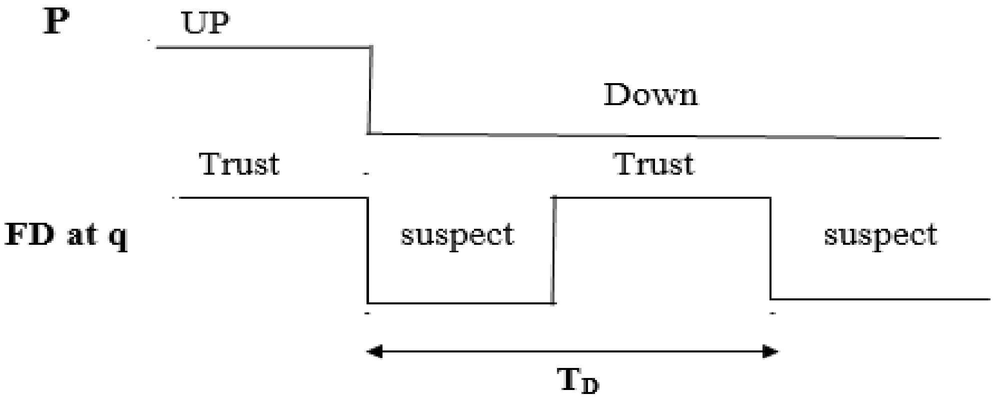


Figure 1.5: Detection Time (TD)

- **Mistake recurrence time ($T_{MR}$):** Time between two successive mistakes.
- **Mistake Duration ($T_M$):** Time taken by a failure detector to correct the mistake. Failure detectors that adapt themselves to the changing network conditions and application requirements are named as adaptive failure Detectors (Hayshibara et al, 2004). Most adaptive failures are based on heartbeat protocol where previous information is used for the prediction of arrival time of next heartbeat.

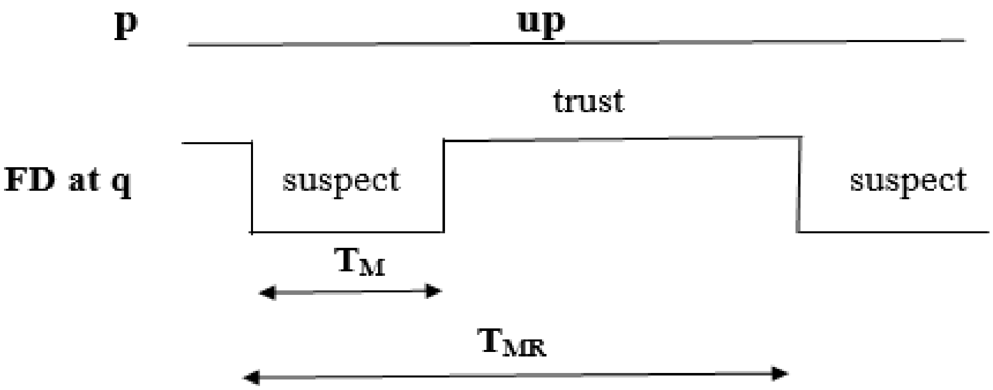


Figure1.6: Mistake duration ($T_M$) and Mistake rate ($T_{MR)}$

### 1.3.2 Heartbeat Strategy for failure detection:

- **Simple heartbeat failure detector Algorithm**

Heartbeat is a widely implemented strategy for failure detectors. After a fixed interval of time every process p send "I am alive" message to a process q. q waits for the message from p till the expiration of timeout from p and if the message is not received it adds p to list of suspected processes. If q later receives "I am alive" message from p, it will remove the process p from list of suspected processes.

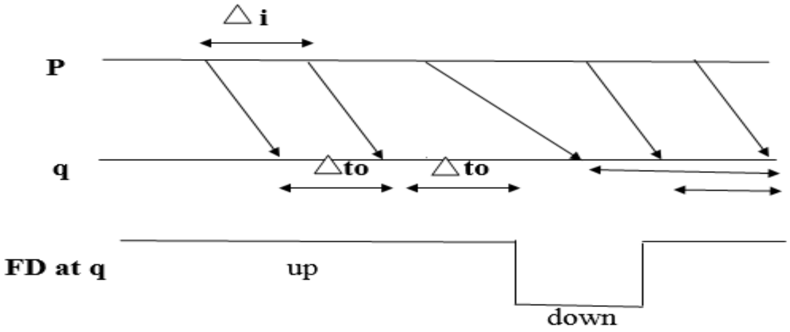


Figure 1.7: Simple heartbeat strategy

The drawbacks of algorithm are related to accuracy and detection time. Consider an ith heartbeat message, mi whose premature timeout should generally depend on mi and its delay. However in the common heartbeat strategy, the probability of premature timeout mi is also dependent on the heartbeat mi-1. Since the timer of mi starts when mi-1 is received and In case mi-1 is fast, the timer for mi will start soon thereby increasing the probability of premature timeout on mi.

In this algorithm the worst case detection time is calculated as the sum of maximum message delay and t0 time units after the crash of process. However this is impractical as in majority of systems, magnitude of maximum message delay is larger than the average message delay.

- **Heartbeat failure detector algorithm with freshness points.**

Chandra and Toug (Chen et al, 2002) proposed an improvement of this classic heartbeat implementation. In the proposed algorithm, the process q (monitoring process) uses a sequence of fixed time points $T_1$, $T_2$, $T_3$ …called freshness points in order to determine whether to suspect the process p. The freshness point $T_i$ is an estimation of arrival time of heartbeat from p.

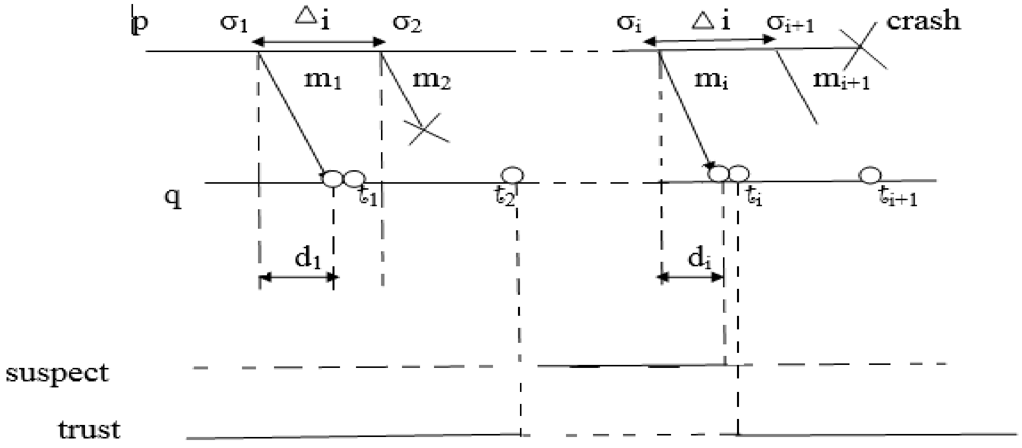


Figure 1.8: Heartbeat with Freshness points

The advantage of this algorithm is that detection time is independent from the last heartbeat message, thus increasing accuracy of the failure detector as it avoids premature timeout.

### 1.3.3 Existing Strategies using Heartbeat

- **Chen FD :**

  Chen et al (2002) proposed an approach that is based on probabilistic enquiry of network traffic. To figure out the estimation of the arrival time of the next heartbeat, Chen FD uses arrival times sampled in the recent past [15]. The expected time is set according to this estimation along with a safety margin and the value is recomputed for every interval.

  The theoretical arrival time EA $_{(K+1)}$ for n messages is given by:

$$EA_{(k+1)} = \frac{1}{n} \sum_{i=k-n-1}^{\kappa} (A_i - \Delta_i * i) + (k+1)\Delta_i,$$

It is assumed that p sends periodic heartbeat messages to q. $m_1$, $m_2$, $m_3$…..are the most recent n heartbeat messages in a sliding window which are considered by process q. $A_1$, $A_2$, $A_3$ …are the actual receiving △ times according to q's clock. $_i$ is the sending interval. The next timeout delay is calculated as sum of EA $_{(K+1)}$ and the constant safety margin a.

$$T_{(K+1) = a +} EA_{(K+1)}$$

- **Bertier FD :**

Bertier introduced a failure detector principally intended for LAN environments. Their proposed algorithm uses the same mechanism as Chen for estimating expected arrival times, but a dynamic way of computing freshness points based on Jacobson's estimation [Bertier et al, 2003]. Bertier FD adapts the safety margin every time it receives a message. The adaptation of the margin *α* is based on the variable *error* in the last estimation. The proposed algorithm is:

$$\begin{gathered}
error_k = A_k - EA_{(k)} - delay_{(k)}, \\
delay_{(k+1)} = delay_{(k)} + \gamma \cdot error_{(k)}, \\
var_{(k+1)} = var_{(k)} + \gamma \cdot (|error_{(k)}| - var_{(k)}), \\
\alpha_{(k+1)} = \beta \cdot delay_{(k+1)} + \phi \cdot var_{(k)}, \\
\tau_{(k+1)} = EA_{(k+1)} + \alpha_{(k+1)},
\end{gathered}$$

In the algorithm, ɤ represents the importance of the new measure with respect to the previous ones. Variable delay is representing estimate margin and *var* is used to estimate magnitude of errors. B and ф are used to adjust variance var.

- **The *φ* FD :**

Instead of providing information having a conventional binary nature i.e. true or suspect, the φ-FD gives a suspicion level on a continuous scale which makes it different from Chen and Bertier- FD [Hayashibra et al, 2004]. In *φ* FD, the suspicion level is given by a value called *φ*, expressed on a scale that is dynamically adjusted to reflect current network conditions (Chen et al, 2002).

Value of ф can be calculated as:

$$\phi(t_{now}) = -lg(P_{later}(t_{now} - T_{last})).$$

Here,

$$P_{later}(t) = \frac{1}{\sigma\sqrt{2\pi}} \int_{t}^{+\infty} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = 1 - F(t),$$

$T_{last}$ is the time when most recent heartbeat was received, $T_{now}$ represents the current time and $p_{later}(t)$ is the probability of arrival of heartbeat more than t time units after the previous one. F(t) is the cumulative distribution function of a normal distribution. An application triggers different actions based on the value of $\phi$.

## 1.4 Overview of Artificial Neural Networks:

The inventor of one of the first neurocomputers, DR. Robert Hecht-Nielsen defines Artificial Neural Networks as: "*a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs*"

ANNS can be defined as electronic models that are based on the neural structure of brain. ANNS are usually presented as a structure of interconnected neurons that resembles the biological neural network. It works on the basic principle of brain that is learning by experience and it learns or changes itself based on the input and output or we can say an ANN is effected by the data that flows through it.

### 1.4.1 Architecture of Neural Networks:

Artificial Neural Networks are generally classified as feed forward and feed backward networks.

- **Feed Forward Artificial Neural Networks:** In Feed forward ANNS the connections between the nodes do not form the directed graphs. They are non-recurrent networks where the signals can only travel in one direction. The information travels in forward direction from input nodes via hidden layer to the output nodes. Each processing layer performs calculations on the input data and the output of each layer is passed as an input to feed the next layer. There are no feedback loops thus the output of any layer cannot effect the same layer. The process of calculation continues until data passes all the layers and an output is calculated.

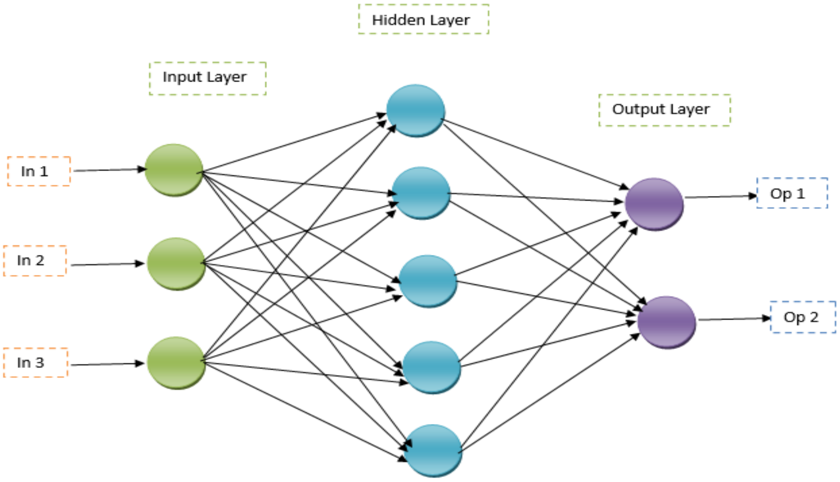


Figure 1.9: Feed Forward Artificial Neural Networks

- **Feed backward Artificial Neural Network:** Feedback networks are non-linear, dynamic systems that modify themselves constantly until the state of equilibrium is achieved .They have feedback loops that allow data to travel in both directions. Any possible connections between neurons is acceptable. They are powerful networks but are extremely complicated. They are also known as recurrent or interactive networks.

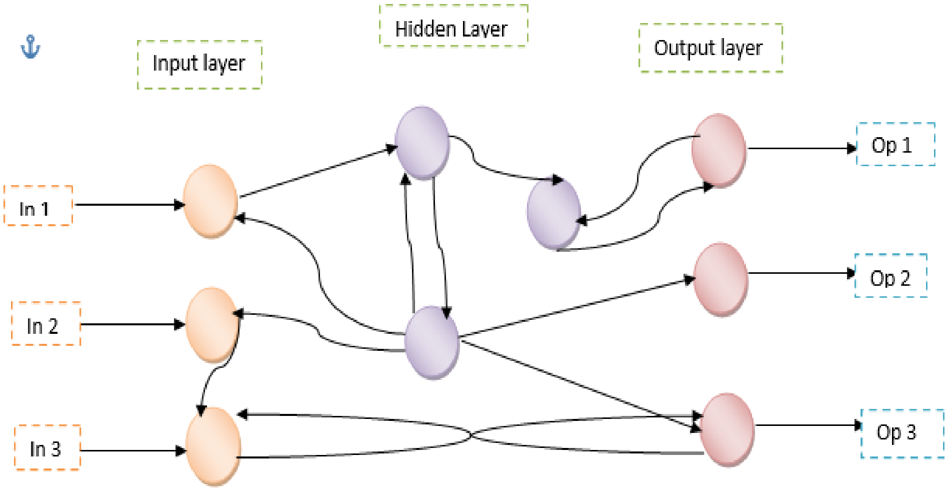


Figure 1.10: Feed Backward Artificial Neural Networks

### 1.4.2 Training an Artificial Neural Network:

Neural Networks can be trained in two ways:

- **Supervised Training:** Both inputs and outputs are provided to the network which processes the inputs and compares the resulting outputs with the target outputs. Errors are calculated and propagated back to the network and accordingly weights are adjusted. This process is repeated several times until the network is trained. Most common algorithms used to propagate errors to hidden layers and adjust weights is back propagation algorithm.
- **Unsupervised Training:** Also known as adaptive training. Here the network is only provide with the inputs .such type of networks do not have any target outputs. The goal of such type of network is group input data.

### 1.4.3 Artificial Neural Networks for Fault tolerance in Clouds

1. Neural networks have the ability to extract meaning from indefinite and complex data and thus can be used for pattern extraction and trend detection which are difficult to be observed by humans or any other computer skills.
2. A trained neural network becomes an expert in the field of information that it analyses and then it can be used for predicting outputs based on given inputs.
3. ANNS have the adaptive learning capability that enables them to perform tasks based on the data given for training or initial experience.
4. The data that is received during the learning time Of ANN is organized and represented by the ANN in its own way.
5. ANNS are fault tolerant due to the redundant information coding property. The network may undergo partial destruction which results in performance degradation. In case of major damage certain network abilities can be retained.
6. ANNS are timesaving and less costly because they take data samples as inputs instead of whole data sets for calculating the output.

# Chapter 2

# LITERATURE REVIEW

This chapter explains in detail survey of research that has contributed in the Cloud Computing environment and fault tolerance in Cloud.

**Alain Tchan et al (2012)** have discussed the Fault tolerance techniques in cloud platforms. They have identified different types of failures as hardware failure, VM failure and application failure. A different kind of strategy is applied at each failure.

**Abhishek Bichawat et al (2011)** proposed a proactive fault tolerance approach for mobile devices. After the allocation of job to a device, it is monitored by a monitor process running in background. While the mobile device is executing the allotted job, the battery level may fall below a threshold, then the monitor process interrupts and directs the job to be migrated .After being interrupted the device establishes a connection with the server and transfers the current state of job to the server which in turn allocates the job to other potential device and guides the device to start from the saved state.

**Anjali Meshram et al. (2013)** proposed fault tolerance model for cloud (FTMC). This model accesses the reliability of computing nodes and choses the node for the computation on the basis of reliability. The node can be removed if it does not perform well. There are n virtual machines which can run n no of different algorithms. The result from each node is transferred to ACCEPTOR which verifies the output. Timings of the result are checked by TIMER after being passed to it and then reliability of each module is calculated on the basis of timing by RELIABILITY ASSESSOR. Finally all results are passed to DECISION MAKER which selects the output on the basis of best reliability.

**Anju Bala et al. (2014)** put forward an idea of designing an intelligent task failure detection models for facilitating proactive fault tolerance by predicting task failures for scientific workflow applications. The working of model is distributed in two modules. In first module task failures are predicted with machine learning approaches and in second module the actual failures are located after executing workflow execution in cloud test-bed. Machine learning approaches such as naïve Bayes, ANN, logistic regression and random forest are implemented to predict the task failures intelligently from the dataset of scientific workflows.

**Cijo George et al (2014)** developed proactive fault tolerance mechanism that avoids significant number of failures. The mechanism is based on partial replication of a set of application processes. An application is started by the framework with a fixed small number of replicated processes. In order to guarantee a strong replica of each failed node, the proposed framework adaptively changes the replicated process set based on failure predictions. The adaptive methodology maintains the advantages of replication while minimizing its limitations. PA$^{R}$EP- MPI allows the changing of replica set by simply copying the process state of a process from local node to a process in the remote node and makes the remote process to act as the replica of the local process.

**Dawei Sun et al. (2013)** put forward a dynamic adaptive fault tolerance strategy (DAFT) that is focused around the standards and semantics of cloud fault tolerance. An analysis on relationship between different failure rates and two different fault tolerance techniques, check-pointing and replication has been carried out. A dynamic adaptive model has been built by combining the two fault tolerance models which helps to increase the serviceability.

**Fabio Lima et al (2004)** proposed adaptive failure detectors that are adjustable to the changing communication loads and use artificial neural networks for predicting the arrival time of next heartbeat from a virtual machine. SNMP (Simple Network Management Protocol) has been used for obtaining training patterns that feed the neural network.

**Hwamin Lee et al. (2009)** proposed a fault tolerant and recovery system called FRAS system (Fault Tolerant and recovery Agent System) for distributed computing systems which provides fault tolerance and recovery mechanisms using agents and is based on  Independent check-pointing and message logging. FRAS is based on four types of agents each having a specific functionality. Recovery agent performs roll back recovery after occurrence of failure. Information agent hypothesis domain knowledge and information during a failure free operation. Facilitator controls the communication between agents and garbage collection agent performs garbage collection of data. Agent recovery algorithm is proposed to maintain a consistent state of a system and prevent domino effect caused due to agent failures. Identifying the dependability of previous rollback-recovery protocols on inherent communication and the underlying operating system, they proposed a rollback recovery protocol that works independently and is more portable and extensible.

**Hiep Nguyen (2013)** proposes that pinpointing a faulty component is the biggest challenge for diagnosing an abnormal distributed application .Black-Box online fault localization system called F-chain has been presented that can pinpoint faulty components immediately after a performance anomaly is detected. F-chain is presented as: a practical online fault localization system for large scale Iaas clouds. This system does not depend upon prior knowledge i.e. previously seen and unseen anomalies, and is practical for Iaas clouds. To achieve higher pinpointing accuracy, an integrated fault localization scheme has been introduced that consider both fault propagation patterns and inter component dependencies.

**Jasbir Kaur et al (2014)** proposed a reactive fault tolerance approach among servers. The work of faulty server is relocated to a new server having minimum load at the time when fault occurs. An algorithm MPI with lookup tables has been proposed and compared to simple MPI.

**Naixue Xiong et al. (2007)** investigates Failure detector properties with connection to real and programmed fault-tolerant cloud based network systems, in order to discover a general non-manual investigation strategy to self-tune corresponding parameters to fulfill user requirements. Based on this general self-tuning method, they propose a dynamic and programmed Self-tuning Failure Detector scheme, called SFD, as an improvement over existing schemes.

**N.R Rejinpaul et al (2012)** proposed a smart checkpoint infrastructure for virtualized service providers, where checkpoints are stored in hadoop distributed file system. Since the checkpoints are distributed in every node of provider, it will enable a task to resume quickly after a nodes crashes. They have demonstrated the infrastructure as an efficient way of making faster checkpoints with minimum interference on task execution.

**Ravi Jawahar et al. (2012)** provided a new dimension for applications deployed in a cloud computing infrastructure which can obtain required fault tolerance properties from a third party. They proposed an innovative perception on handling fault tolerance in order to achieve reliability. With the help of this methodology user can specifically apply the desired level of fault tolerance without any knowledge of implementation. The model straightforwardly work fault tolerance solution to user's applications by combining selective fault tolerance mechanisms and discovers the properties of a fault tolerance solution by method of runtime monitoring.

**Shun-Sheng et al (2010)** proposed Dual Agreement Protocol of Cloud Computing (DAPCC), keeping in consideration the scalable and virtual nature of cloud. DAPCC is proposed to tackle the agreement problem caused by faulty nodes which send wrong messages, it tells how the system achieves agreement in a cloud computing environment. Using minimal number of message exchange rounds this protocol can tolerate large number of allowable faulty nodes.

**Shivam Nagpal et al (2013)** proposed a fault tolerant model that takes decisions on the basis of reliability of nodes/virtual machines. Reliability is estimated on the basis of 2 parameters; accuracy and time and it increases if correct result is produced in time. If any of the nodes does not achieve the level then backward recovery is performed by the system. This model focuses on adaptive behavior of processing nodes and the nodes are removed or added on the basis of reliability.

**Sagar C Joshi et al (2014)** proposed a fault tolerance mechanism to handle server failures by migrating the virtual machines hosted on the failed server to a new location. Virtualization has been applied for data centers giving rise to the concept of virtual Data Centers (VDC) which have virtual Machine (VM) as the basic unit of allocation. Using appropriate resource allocation algorithms, multiple VDCs can be hosted on a physical data center. A load balancing scheme, based on clustering has been presented which reduces the impact of server failures on VDC that are hosted in data center. It allocates the VDC requests evenly across the data center network.

**Shewta Jain et al (2014)** proposed Fault detection and tolerant system (FDTS) which detects the faulty application by heartbeat and gossip algorithm. If an application is identified as faulty, then the FDTS deploys the application recovery mechanism at saas layer. In order to guarantee users with smooth functioning of applications and minimum downtime, this mechanism will start, recover or reinstall the faulty application.

**Wenbing Zhao et al. (2010)** proposed Low Latency Fault Tolerance (LLFT) Model that utilizes leader/follower replication approach and provides fault tolerance for distributed applications deployed within a cloud computing environment. The novel commitments of the LLFT middleware incorporate the low Latency Messaging Protocol, the leader-determined membership protocol and the virtual determinizer Framework. LLFT middleware protects the application against faults by maintaining its replicas without any modifications in the original application.

# Chapter 3
# PRESENT WORK

This chapter contains the Research problem, objectives and research methodology.

## 3.1 Problem Formulation:

As per as the research gaps analyzed there is a potential need for implementing autonomic fault tolerance by using different parameters in cloud environment. During the literature review the various challenges faced by academicians in incorporating fault tolerance in cloud computing is as follows:

- The heterogeneity of the cloud is the biggest hindrance to localize the faults. There is a need to implement efficient techniques for locating the faults.
- There are more chances of errors because processing is done on remote computers.
- Failures occurring in the data centers are not in the scope of the user's organization necessitating the implementation of an autonomous fault tolerance technique for applications computing on cloud environment.
- It is difficult to interpret the changing system state because cloud environment are dynamically scalable, unexpected and often virtualized resources are provided as a service.
- Limited information is provided to the users because of high system complexity, so it is difficult to design an optimal fault tolerance solution.
- Fault Prediction and Monitoring framework needs to be developed for real time applications that execute in cloud environment.

## 3.2 Objective:

1. Analysis of various fault tolerance and fault detection techniques in cloud computing.
2. Analysis of various models of Artificial Neural Networks used for fault detection.
3. Designing proactive Fault tolerance approach in cloud computing using Artificial Neural Networks.
4. Verify the proposed approach by simulating in cloud environment.

## 3.3 Research Methodology:

### 3.3.1 Model used

Multilayer feed forward neural network is used to predict the virtual machine (VM) number and heartbeat. Our model consists of one input layer, one hidden layer and one output layer. The inputs to input layer are values of previous observed target VM-number and heartbeat represented by I1 and I2 respectively. The outputs of the network are the prediction for VM number and the next heartbeat symbolized by O1 and O2 respectively.

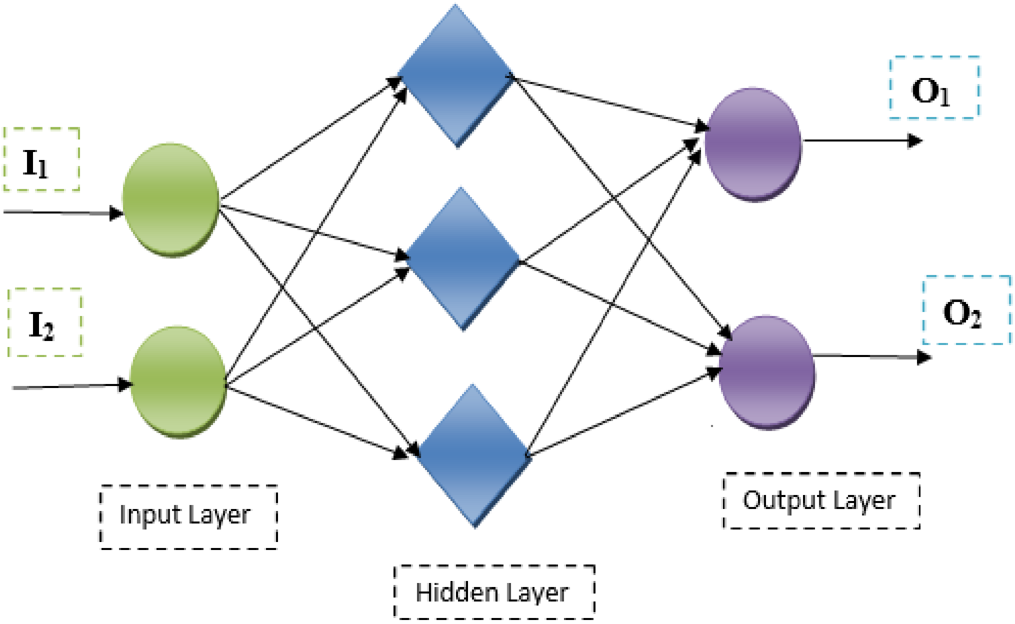


Figure 11: Multi-Layer Feed Forward Network

### 3.3.2 Training of Network:

Multilayer feed forward network is trained in supervised manner. Here the network is provided with inputs and target outputs. After processing the inputs and calculation of outputs, they are compared with target outputs. Errors are calculated and propagated back to the network and accordingly weights are adjusted. This process is repeated several times until the network is trained. The algorithm used here to propagate errors to hidden layers and adjust weights is back propagation algorithm.

Back propagation algorithm generally learns by examples. We give the network examples of desired outputs and it accordingly adjusts its weights and after completion of training it gives the preferred outputs for specific inputs. It is necessary to provide the network with the examples of expected outputs called target.

The first step in training the network is initializing the network by fixing all the weights to small random numbers such as -1 to 1. In the next step the input pattern is applied and the output is generated, this is called forward pass. Since the weights have been applied randomly thus the output will be completely different from the target output. Then error of each output neuron is calculated as target-output. In reverse pass, this error is used to propagate changes in the weights so that the error is reduced or output of every neuron is close to the target. This procedure is repeated again and again to get a minimum error.

The training process can stopped when the system is fully trained or no further learning is required. In practice it is stopped when the error reaches to a minimum value or some predetermined value. When finally our network gets trained, the training process is stopped and weights are fixed.

**Back propagation algorithm:**

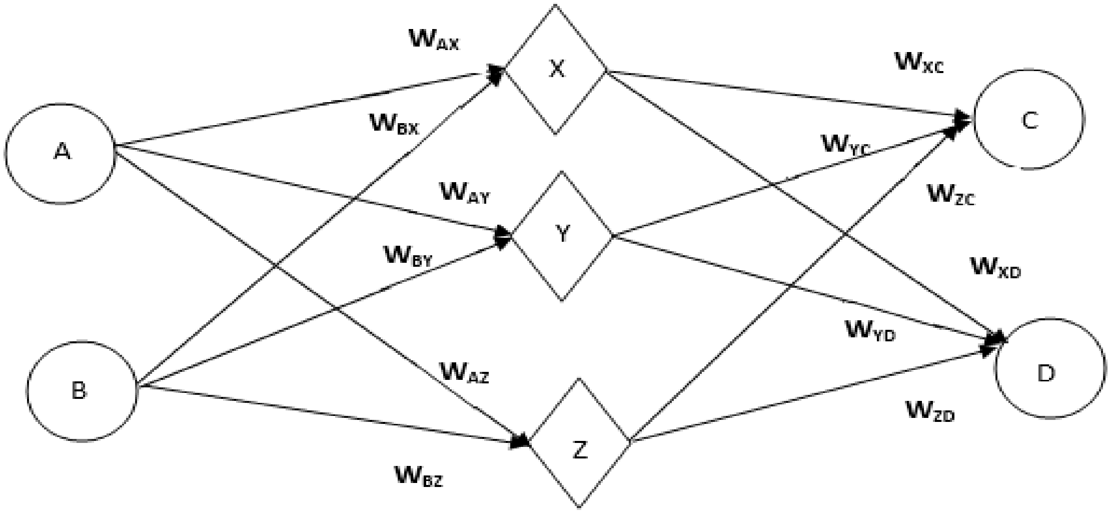


Figure 3.2: Back Propagation Algorithm

**Step 1:** Apply random inputs to the network and calculate the output. The output generated will be any value because of the random inputs.

**Step 2:** Calculate the error for output neurons.

$$Error_{C} = output_{C} (1-output_{C}) (Target_{C} – Output_{C})$$

$$Error_{D} = output_{D} (1-output_{D}) (Target_{D} – Output_{D})$$

**[**Output $_X$ (1-output $_X$) is used because of sigmoid function, in case of threshold neuron error is simply calculated as:

$$Error_{X} = (Target_{X} – Output_{X})]$$

**Step 3**: Adjust the weights of output layer.

$W_{XC}^{+} = W_{XC} + n (Error_{C} * output_{X})$ $W_{XD}^{+} = W_{XC} + n (Error_{D} * output_{X})$

$W_{YC}^{+} = W_{YC} + n (Error_{C} * output_{Y})$ $W_{YD}^{+} = W_{YD} + n (Error_{D} * output_{Y})$

$W_{ZC}^{+} = W_{ZC} + n (Error_{C} * output_{Z})$ $W_{ZD}^{+} = W_{ZD} + n (Error_{D} * output_{Z})$

**Step 4**: Calculate hidden layer errors by back propagation

$$\text{Error}_{X} = \text{output}_{X}\ (1\text{-output}_{X})\ (\text{Error}_{C} * W_{YC} + \text{Error}_{D} * W_{XD})$$

$$\text{Error}_{Y} = \text{output}_{Y}\ (1\text{-output}_{Y})\ (\text{Error}_{C} * W_{XC} + \text{Error}_{D} * W_{YD})$$

$$\text{Error}_{Z} = \text{output}_{X}\ (1\text{-output}_{X})\ (\text{Error}_{C} * W_{ZC} + \text{Error}_{D} * W_{ZD})$$

**Step 5:** Adjust the weights of hidden layer

$W_{AX}^{+} = W_{AX\,+}\ n\ (\text{Error}_{X} * \text{Input A})$ $W_{BX}^{+} = W_{BX\,+}\ n\ (\text{Error}_{X} * \text{Input B})$

$W_{AY}^{+} = W_{AY\,+}\ n\ (\text{Error}_{Y} * \text{Input A})$ $W_{BY}^{+} = W_{BY\,+}\ n\ (\text{Error}_{Y} * \text{Input B})$

$W_{AZ}^{+} = W_{AZ\,+}\ n\ (\text{Error}_{Z} * \text{Input A})$ $W_{BZ}^{+} = W_{BZ\,+}\ n\ (\text{Error}_{Z} * \text{Input B})$

[n is the learning rate generally equal to 1]

## 3.4 Tools used:

**Cloudsim:** cloudsim toolkit is used for simulating cloud computing scenarios. It can be defined as a generalized and extensible framework which supports smooth modelling and simulation of application performance. It enables users to concentrate on the particular issues that they want to examine without having the knowledge of cloud infrastructure or the services. Cloud designers can test the performance of their strategies without any cost in a manageable and scalable environment. The simulator does not run any actual software, it is a running model of an environment embedded in hardware model that provides abstraction of technology related details.

Cloudsim provides users with a library of essential classes used for describing data centers, virtual machines, applications, users, computational resources and various policies for managing different parts of the system. These components enable a user to evaluate new policies and evaluate their competence in terms of cost, application and execution time. By simply writing a java program it enables users to add a desired scenario.

To set up the development environment we need java development kit (JDK) and eclipse (IDE).We have used java SE Development kit and eclipse luna.

JDK is a software development environment that is used for developing java applications and applets. It is an assembly of java Runtime environment, an interpreter/loader, compiler (javac), an archiver (jar), documentation generator and various other tools required in java development. Java SE uses object oriented java programming language. It is a part of java software platform family.

Eclipse an integrated development environment provides users with a base workspace and plug-in system for modifying the environment. Eclipse luna includes an official support of java 8 in java development tools, object teams, eclipse communication framework, web tools platform, plugin development tools, memory analyzer etc.

# Chapter 4

# RESULTS AND DISCUSSIONS

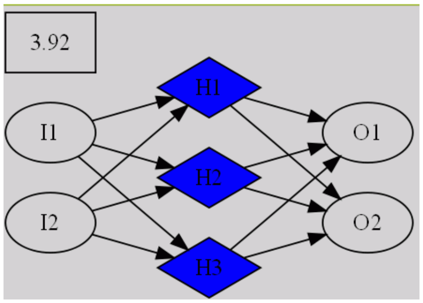
In this chapter, the experimental results of proposed approach are discussed. Cloud Sim is used for simulating the cloud environment.

The Multilayer feed forward network is trained using back propagation algorithm to predict the VM number and the next heartbeat. The activation function used here is sigmoid function so the data is normalized in the range of [-1, 1]. Both input and output data is normalized for better results and less calculation time.

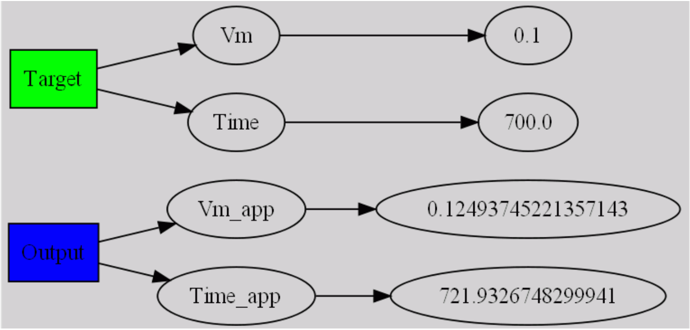
We have considered 4 virtual machines in this scenario, each VM will detect the VM number from which it will receive the heartbeat and the time-unit after which heartbeat /ping will be received.

Figures a.1 to a.7 represent the state of the neural network at random time intervals. Error rare calculated is also shown in the figure.

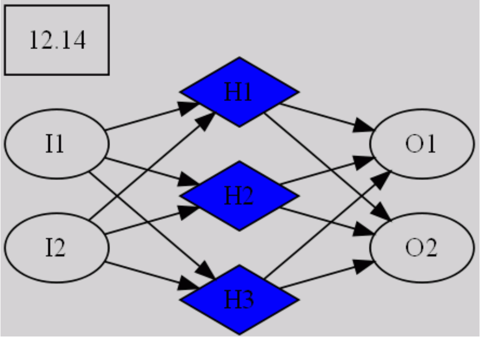
In Figures b.1 to b.7 the outputs of neural network i.e. vm_app and Time_app are presented along with the actual/target outputs.



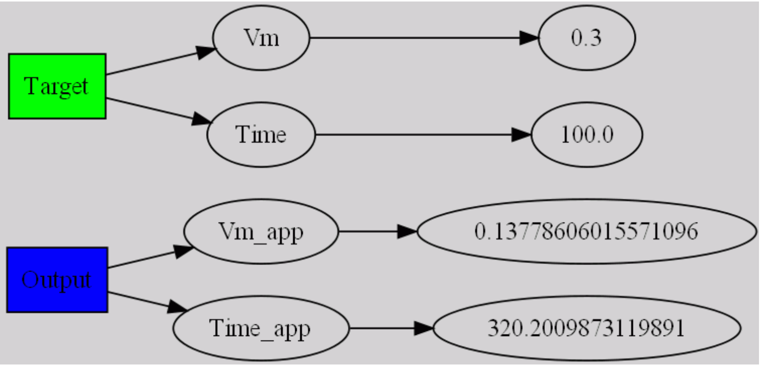
a.1



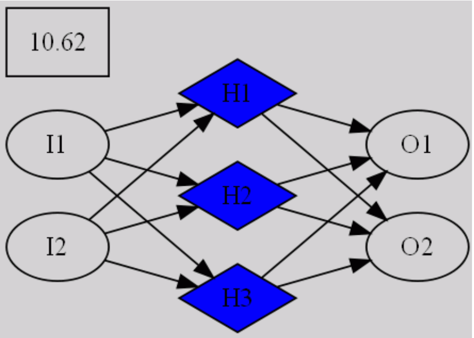
b.1



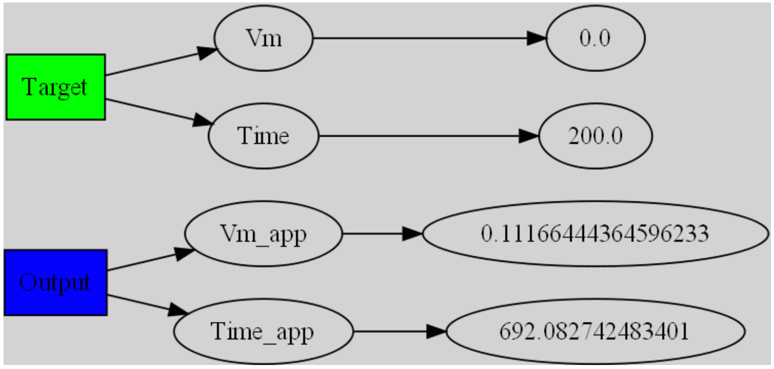
a.2



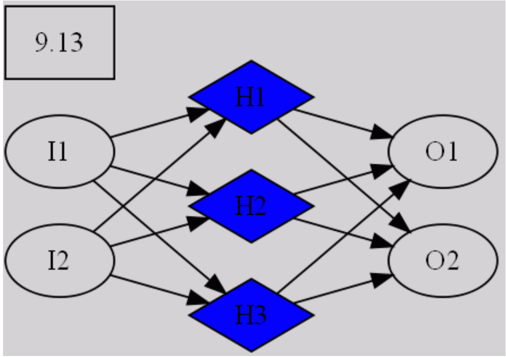
b.2



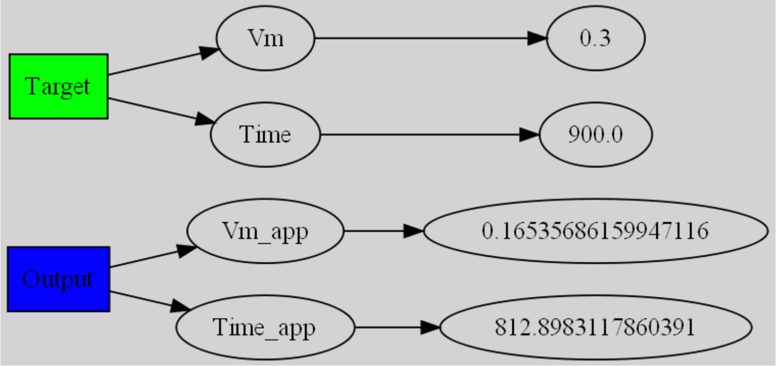
a.3



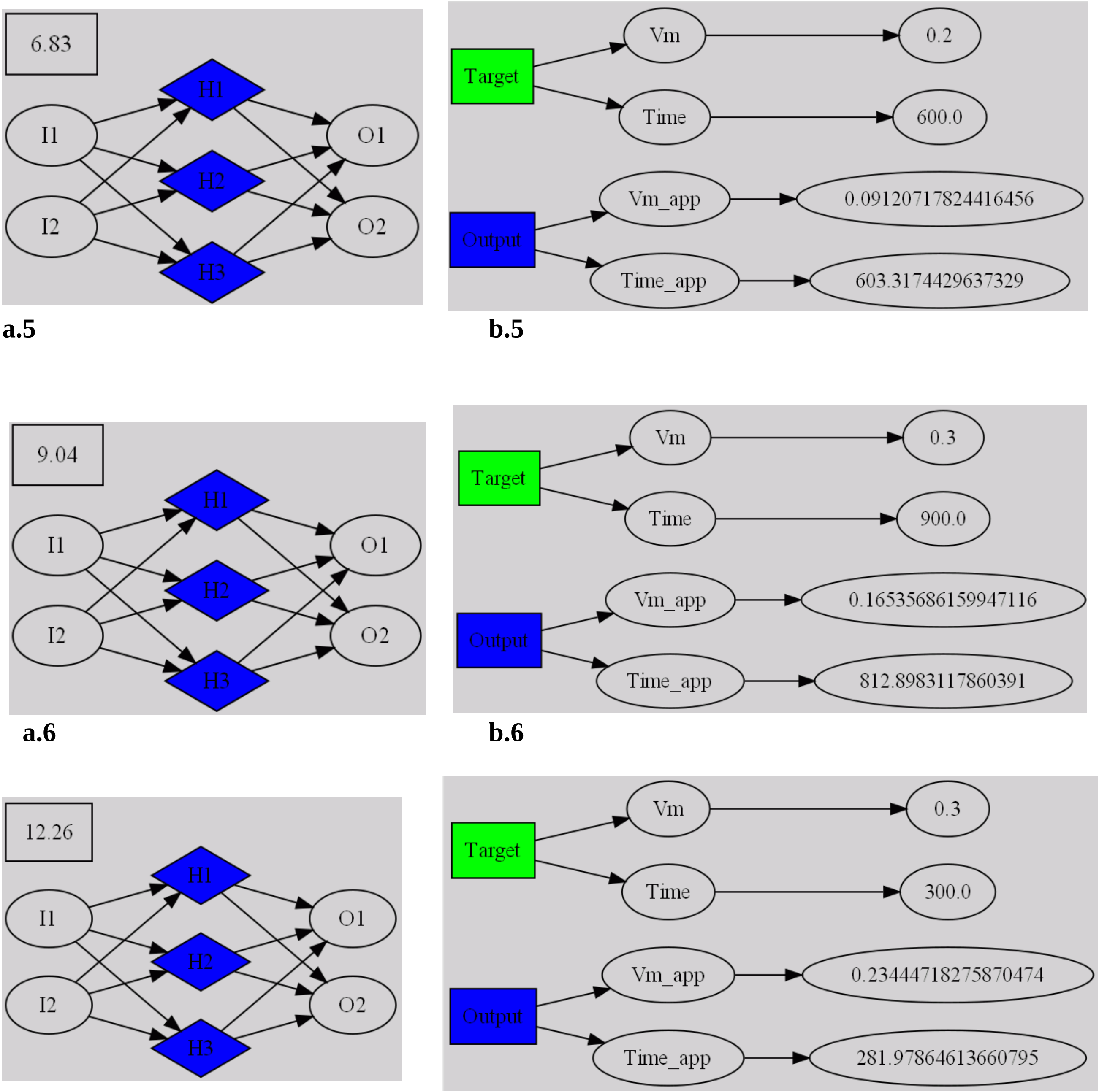
b.3



a.4



b.4



**a.5** **b.5**

**a.6** **b.6**

**a.7** **b.7**

Figure 4.1.a) Error Calculation in neural network

b) Output of Neural Network compared with target

| S.no | Target Heart-ping | Output Heart-ping |
|---|---|---|
| 1 | 0.7 | 0.727 |
| 2 | 0.8 | 0.811 |
| 3 | 0.1 | 0.171 |
| 4 | 0.4 | 0.389 |
| 5 | 0.8 | 0.810 |
| 6 | 0.9 | 0.844 |
| 7 | 0.1 | 0.084 |
| 8 | 0.2 | 0.148 |
| 9 | 0.6 | 0.619 |
| 10 | 0.8 | 0.811 |
| 11 | 0.3 | 0.293 |
| 12 | 0.4 | 0.390 |
| 13 | 0.5 | 0.510 |
| 14 | 0.7 | 0.732 |
| 15 | 0.0 | 0.084 |
| 16 | 0.5 | 0.509 |
| 17 | 0.6 | 0.622 |
| 18 | 0.1 | 0.092 |

Table 4.1: Values of target heart-ping and output heart-ping at random intervals.

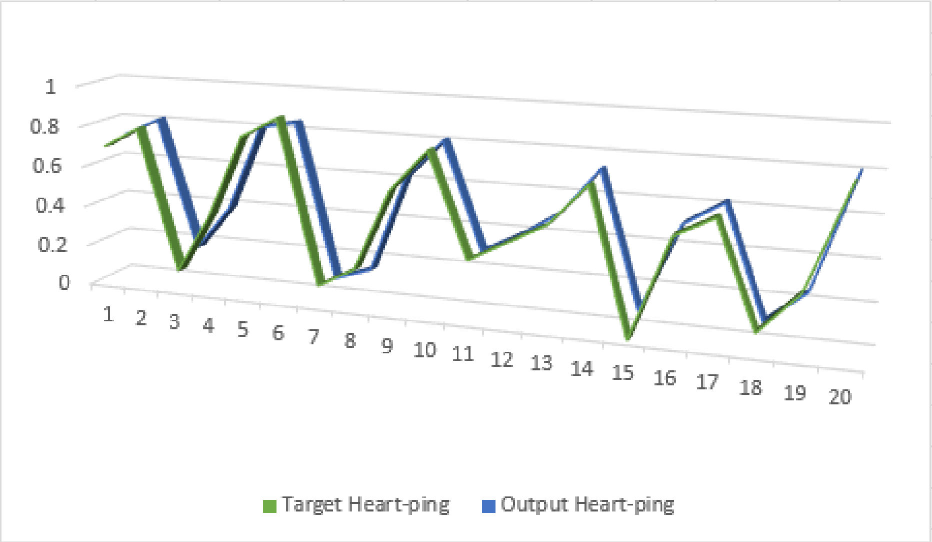


Figure 4.2: Comparison graph for Target Heart-ping and output heart-ping.

| S.no | Target VM | Output VM |
|---|---|---|
| 1 | 0.000 | 0.128 |
| 2 | 0.100 | 0.120 |
| 3 | 0.000 | 0.140 |
| 4 | 0.300 | 0.293 |
| 5 | 0.100 | 0.101 |
| 6 | 0.000 | 0.101 |
| 7 | 0.100 | 0.099 |
| 8 | 0.300 | 0.282 |
| 9 | 0.100 | 0.103 |
| 10 | 0.300 | 0.305 |
| 11 | 0.300 | 0.293 |
| 12 | 0.100 | 0.103 |
| 13 | 0.100 | 0.108 |
| 14 | 0.300 | 0.287 |
| 15 | 0.100 | 0.171 |
| 16 | 0.300 | 0.293 |
| 17 | 0.300 | 0.282 |
| 18 | 0.000 | 0.133 |

Table 4.2: Values of Target VM and output VM at random Intervals

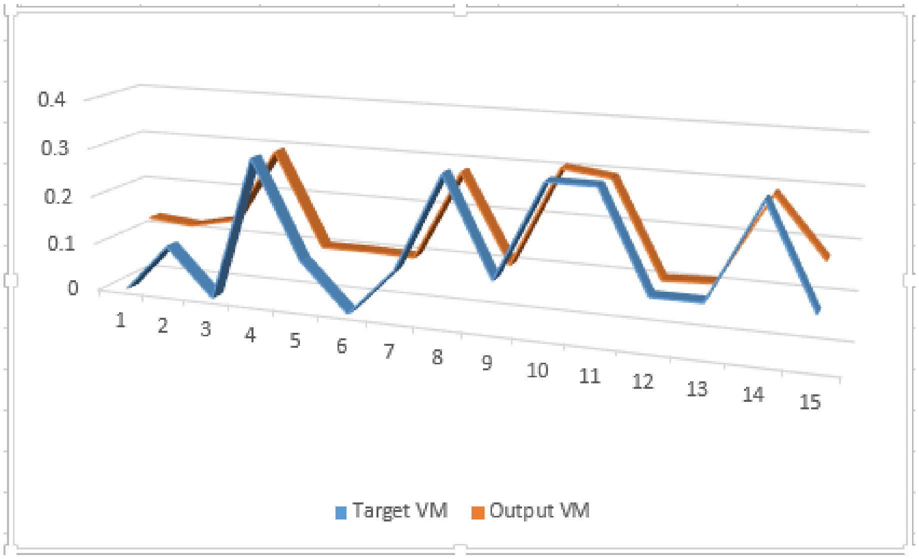


Figure 4.3: Comparison graph for target VM and output VM

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion:

Cloud environment is dynamic which leads to unexpected system behavior resulting in faults and failures. In order to improve reliability and achieve robustness in cloud computing, failures should be assessed and handled effectively. Fault detection is one of the biggest challenges in making a system fault tolerant.

This thesis aims in increasing the fault tolerance of Cloud by detecting the faults proactively with the help of artificial neural networks.

## 5.2 Future scope:

In this research Multilayer Perceptron Neural Network (MLPNN) with one hidden layer has been used. The research can be further extended by using more than one hidden layers in the Neural Network.

# Chapter 6

# REFERENCES

A Vouk, M. (2008). Cloud computing–issues, research and implementations. CIT. Journal of Computing and Information Technology, 16(4), 235-246.

Bahl, P., Chandra, R., Greenberg, A., Kandula, S., Maltz, D. A., & Zhang, M. (2007, August). Towards highly reliable enterprise network services via inference of multi-level dependencies. In ACM SIGCOMM Computer Communication Review (Vol. 37, No. 4, pp. 13-24). ACM.

Bala, A., & Chana, I. (2014). Intelligent failure prediction models for scientific workflows. Expert Systems with Applications.

Bertier, M., Marin, O., & Sens, P. (2002). Implementation and performance evaluation of an adaptable failure detector. In Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on (pp. 354-363). IEEE.

Bertier, M., Marin, O., & Sens, P. (2003, June). Performance analysis of a hierarchical failure detector. In 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 635-635). IEEE Computer Society.

Brian, O., Brunschwiler, T., Dill, H., Christ, H., Falsafi, B., Fischer, M., & Zollinger, M. (2012). Cloud Computing. White Paper SATW.

Carolan, S. J. (2009). Introduction to cloud computing. Architecture. White Paper, 1st edn. Sun Microsystems (June 2009).

Chandra, T. D., & Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. Journal of the ACM (JACM), 43(2), 225-267

Chen, W., Toueg, S., & Aguilera, M. K. (2002). On the quality of service of failure detectors. Computers, IEEE Transactions on, 51(5), 561-580.

Défago, X., Urbán, P., Hayashibara, N., & Katayama, T. (2005, March). Definition and specification of accrual failure detectors. In Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on (pp. 206-215). IEEE.

Deng, J., Huang, S. H., Han, Y. S., & Deng, J. H. (2010, December). Fault-tolerant and reliable computation in cloud computing. In GLOBECOM Workshops (GC Wkshps), 2010 IEEE (pp. 1601-1605). IEEE.

de Araújo Macêdo, R., & e Lima, F. R. L. (2004). Improving the quality of service of failure detectors with SNMP and artificial neural networks. In Anais do 22o. Simpósio Brasileiro de Redes de Computadores (pp. 583-586)

Furht, B. (2010). Cloud computing fundamentals. In Handbook of cloud computing (pp. 3-19). Springer US.

Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008, November). Cloud computing and grid computing 360-degree compared. In Grid Computing Environments Workshop, 2008. GCE'08 (pp. 1-10). Ieee.

Gong, C., Liu, J., Zhang, Q., Chen, H., & Gong, Z. (2010, September). The characteristics of cloud computing. In Parallel Processing Workshops (ICPPW), 2010 39th International Conference on (pp. 275-279). IEEE.

Gupta, I., Chandra, T. D., & Goldszmidt, G. S. (2001, August). On scalable and efficient distributed failure detectors. In Proceedings of the twentieth annual ACM symposium on Principles of distributed computing (pp. 170-179). ACM

George, C., & Vadhiyar, S. Fault Tolerance on Large Scale Systems using Adaptive Process Replication.

Hayashibara, N., Defago, X., Yared, R., & Katayama, T. (2004, October). The φ accrual failure detector. In Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on (pp. 66-78). IEEE.

Huth, A., & Cebula, J. (2011). The Basics of Cloud Computing. United States Computer.

Jhawar, R., Piuri, V., & Santambrogio, M. (2013). Fault tolerance management in cloud computing: A system-level perspective. Systems Journal, IEEE, 7(2), 288-297.

Joshi, S. C., & Sivalingam, K. M. (2014). Fault tolerance mechanisms for virtual data center architectures. Photonic Network Communications, 28(2), 154-164.

Kaushal, V., & Bala, A. (2011). Autonomic fault tolerance using haproxy in cloud environment. Int. J. of Advanced Engineering Sciences and Technologies, 7(2), 54-59.

Malik, S., & Huet, F. (2011, July). Adaptive Fault Tolerance in Real Time Cloud Computing. In Services (SERVICES), 2011 IEEE World Congress on (pp. 280-287). IEEE.

Maier, G., Sommer, R., Dreger, H., Feldmann, A., Paxson, V., & Schneider, F. (2008, August). Enriching network security analysis with time travel. In ACM SIGCOMM Computer Communication Review (Vol. 38, No. 4, pp. 183-194). ACM.

Mell, P., & Grance, T. (2009). The NIST definition of cloud computing. National Institute of Standards and Technology, 53(6), 50.

Meshram, A. D., Sambare, A. S., & Zade, S. D. (2013). Fault Tolerance Model for Reliable Cloud Computing

Nguyen, H., Shen, Z., Tan, Y., & Gu, X. (2013, July). FChain: Toward black-box online fault localization for cloud systems. In Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on (pp. 21-30). IEEE.

Patra, P. K., Singh, H., & Singh, G. (2013). Fault Tolerance Techniques and Comparative Implementation in Cloud Computing. International Journal of Computer Applications, 64(14).

Plummer, D. C., Cearley, D. W., & Smith, D. M. (2008). Cloud computing confusion leads to opportunity. Gartner Report

Sun, D. W., Chang, G. R., Gao, S., Jin, L. Z., & Wang, X. W. (2012). Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. Journal of computer science and technology, 27(2), 256-272.

Tchana, A., Broto, L., & Hagimont, D. (2012, March). Fault Tolerant Approaches in Cloud Computing Infrastructures. In ICAS 2012, The Eighth International Conference on Autonomic and Autonomous Systems (pp. 42-48).

Wang, S. S., Yan, K. Q., & Wang, S. C. (2011). Achieving efficient agreement within a dual-failure cloud-computing environment. Expert Systems with Applications, 38(1), 906-915.

Xiong, N., Vasilakos, A. V., Yang, Y. R., Qiao, C., & Andy, Y. P. (2012). A class of practical self-tuning failure detection schemes for cloud communication networks. IEEE/ACM Transactions on Networking (ToN), submitted.

Youssef, A. E. (2012). Exploring Cloud Computing Services and Applications.Journal of Emerging Trends in Computing and Information Sciences, 3(6), 838-847.

Zhao, W., Melliar-Smith, P. M., & Moser, L. E. (2010, July). Fault tlerance middleware for cloud computing. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 67-74). IEEE.

# LIST OF PAPERS

1. Zeeshan Amin, Harshpreet Singh and Nisha Sethi. Article: Review on Fault Tolerance Techniques in Cloud Computing. *International Journal of Computer Applications* 116(18):11-17, April 2015.
2. Zeeshan Amin, Harshpreet Singh. Article: Proactive Fault Tolerance in Cloud using Multilayer feed forward Artificial Neural Network. Australian journal of basic and applied sciences (communicated).