A Load Balancing Model Based on Cloud Partitioning For the Public Cloud

Enrollment No. : 101264

Name of Student: Sabhyata

Supervisor's Name: Prof. Dr. Deepak Dahiya(CSE and ICT DEPT)

May-2014

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN (H.P)

(i)

Table of Contents

CERTIFICATE

This is to certify that the work titled "**A Load Balancing model based on cloud partitioning for the public cloud "** submitted by **Sabhyata** in partial fulfillment for the award of degree of B.Tech Computer Science Engineering of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

____

(Signature of Supervisor)

Name of Supervisor: Prof. Dr. Deepak Dahiya

Designation: Professor, Dept. of CSE and ICT

Date:

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to Prof. Deepak Dahiya, my Project Guide, for guiding and correcting me at every step of my work with attention and care. He has taken pain to go through the project and make necessary correction as and when needed.

Sincere thanks to **Brig(retd.) S.P Gharera , HOD, CSE & IT Department**, for being cooperative to the students of the department and providing relevant guidance in their endeavors.

I would also like to express my gratitude to this alma mater **JUIT, Waknaghat** for providing proper resources as and when required such as an all time internet facility and other resources.

Hence without giving a warm thanks to all of them who made this project work a reality my work would be incomplete.

Signature of the Student……………………………….

Name of the Student – Sabhyata

Date -

III

SUMMARY

Objective

To develop a better load balance model for public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. This algorithm applies the game theory to the load balancing strategy to improves the efficiency in the public cloud environment .

Existing System

Cloud computing is efficient and scalable but maintaining the stability of processing so many jobs in cloud computing environment is a very complex problem with load balancing receiving much attentions for researchers . Since the job arrival pattern is not predictable and the capacities of each node in cloud differ , load balancing problem , Workload control is crucial to improve system performance and maintain stability . Load balancing Schemes depend on whether the system dynamics are important can be either static and dynamic. Static schemes do not use the system but can change as the system status changes. A dynamic scheme is used here for its flexibility.

Disadvantages :

Load balancing schemes depending on whether the system dynamics are important can be either static and dynamic . Static schemes do not use the system information and are less complex .

List of Figures

Chapter1 : INTRODUCTION

Chapter 1: INTRODUCTION

1.1 Abstract

Cloud computing is an attracting technology in the field of computer science. In Gartner's report, it says that the cloud will bring changes to the IT industry. The cloud is changing our life by providing users with new types of services. Users get service from a cloud without paying attention to the details. NIST gave a definition of cloud computing as 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. More and more people pay attention to cloud computing. Cloud computing is efficient and scalable but maintaining the stability of processing so many jobs in the cloud computing environment is a very complex problem with load balancing receiving much attention for researchers. Since the job arrival pattern is not predictable and the capacities of each node in the cloud differ, for load balancing problem, workload control is crucial to improve system performance and maintain stability. Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic. Static schemes do not use the system information and are less complex while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic scheme is used here for its flexibility. The model has a main controller and balancers to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. The system status then provides a basis for choosing the right load balancing strategy.

The load balancing model given in this article is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for

arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy.

1.2 Problem Statement

What will be done in the project?

Load balancing schemes depending on whether the system dynamics are important can be either static and dynamic . Static schemes do not use the system information and are less complex while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic scheme is used here for its flexibility. The model has a main controller and balancers to gather and analyze the information. Thus, the dynamic control has little influence on the other working nodes. The system status then provides a basis for choosing the right load balancing strategy.

The load balancing model given in this article is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy.

Chapter2 : LITERATURE REVIEW

2.1 Background

There have been many studies of load balancing for the cloud environment. Load balancing in cloud computing was described in a white paper written by Adler who introduced the tools and techniques commonly used for IEEE TRANSACTIONS ON CLOUD COMPUTING YEAR 2013 load balancing in the cloud. However, load balancing in the cloud is still a new problem that needs new architectures to adapt to many changes. Chaczko et al. described the role that load balancing plays in improving the performance and maintaining stability. There are many load balancing algorithms, such as Round Robin, Equally Spread Current Execution Algorithm, and Ant Colony algorithm. Nishant et al. used the ant colony optimization method in nodes load balancing. Randles et al. gave a compared analysis of some algorithms in cloud computing by checking the performance time and cost. They concluded that the ESCE algorithm and throttled algorithm are better than the Round Robin algorithm. Some of the classical load balancing methods are similar to the allocation method in the operating system, for example, the Round Robin algorithm and the First Come First Served (FCFS) rules. The Round Robin algorithm is used here because it is fairly simple.

Cloud computing is causing a transformational shift that touches almost every part of technology landscape. The cloud is changing our life by providing users with new type of services. In Gartner’s report , it says that the cloud will bring changes to the IT industry. Cloud computing is a type of computing in which resources are provided over internet. We can compare cloud model with managed-hosting model where infrastructure is hosted and managed by third party and consumed by users. In the same way, cloud subscribed users consume services which are managed by cloud provider. It is model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources. With this we can provision or release resources with minimum management and service provider interaction. Cloud computing is mostly used to sell hosted services in the sense of application service provisioning that run client server software at remote location. These services are IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service). End users access cloud-based applications through a web browser, thin client or mobile app while the business software and user's data are stored at remote location. IaaS provides virtual machines, servers, storage space, network, load balancers etc. Multiple virtual machines can be built on a single physical server using hypervisor. Each virtual machine has its own operating system. In this model cloud user maintains operating systems and application software. PaaS provides a computing platform including programming language execution environment, database and web server. Using SaaS, users are provided access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications.

Study

1. Infinite numbers of pooled computing resources are available to subscribed users. Cloud computing utilizes pooled computing resources that may be externally purchased and controlled or may be internal resources that are pooled but not dedicated. These resources contributing to the cloud are available to any subscribing users.
2. Virtualization of computer resources.
Virtualization is vital to the cloud because the scale of cloud infrastructure has to be enormous. Each server takes up physical space and significant power and cooling. Hence, getting high utilization from each server is vital to be cost effective.

2.2 Scope of the Proposed System

There are several cloud computing categories with this work focused on a public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider. A large public cloud will include many nodes and the nodes in different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations.
Good load balance will improve the performance of the entire cloud. Better load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations.The current model integrates several methods and switches between the load balance methods based on the system status. A relatively simple method can be used for the partition

idle state with a more complex method for the normal state. The load balancers then switch methods as the status changes. Here, the idle status uses an improved Round Robin algorithm while the normal status uses a game theory based load balancing strategy.

.

Chapter3 : ANALYSIS, DESIGN AND MODELING

3.1 SYSTEM ANALYSIS

Analysis

Introduction to System Analysis

Analysis is the process of understanding the problem and its domain. The main objective of analysis is to capture a complete, unambiguous and consistent picture of the requirements of the system and what the system must do to satisfy the user needs and requirements. The principal objective of the systems analysis phase is the specification of what the system is required to do. The systems development data input and output forms and conventions.

The eventual goal of information systems engineering is to develop software “factories” that use natural language and artificial intelligence technique as part of an integrated set of tools to support the analysis and design of large information

Feasibility Study

The prime focus of the feasibility study is evaluating the practicality of the proposed system keeping in mind a number of factors [9]. The following factors are taken into account before deciding in favor of the new system:

Economic Feasibility

The proposed cellular bidding System will save lots of paper work and Facilitate magnetic record keeping thereby reducing the costs incurred on above heads. This reduction in cost prompts the company to go for such computer-based system.

Technical Feasibility

As the saying goes, "to err is human". Keeping in view the above fact, now days all organizations are automating the repetitive and monotonous works done by humans as well as proving a means of establishing a connection through their j2me enabled mobiles. The key process areas of current system are nicely amenable to automation and hence the technical feasibility is proved beyond doubt.

Operational Feasibility

Since the inception of Internet, it is continuously growing in leaps and bounds, as more people are arriving at the conclusion that they cannot escape from it and in order to keep up, they have to participate in that. In today's world, Cellular Mobile Devices provides an efficient, reliable, cost-effective and timesaving platform for communication. Here in the proposed system bidding is implemented on the mobile devices, which will be enable to end users of Cellular Mobile Devices to obtain the information about the various products that are the part of the system and the at the same time can proceed with the bidding.

Time and Resource Feasibility

This system helps the user to find in the best usage of resources keeping in track of all the product details over a period of time, thereby reducing the decision making process easier and worth while. This acts to be a solution provider in determining the best allocation of resources and finding out the way for time reduction.

3.2FUNCTIONAL REQUIREMENTS

The Functional Requirements Specification documents the operations and activities that a system must be able to perform. a description of the facility or feature required. Functional requirements deal with what the system should do or provide for users. They include description of the required functions, outlines of associated reports or online queries, and details of data to be held in the system.

Each requirement can be perceived as a module or component, each module has specific functionality on which other modules depend on. The module can be part of one software product or can be distributed among several software products and secure communication is established between the different modules.

So here in this system there are four modules .The functional requirement of the system are divided into four modules user, system model, main controller and balancer and cloud partition and load balancing. The functional requirement of the first module is only registered user can access the data in the second model it checks the status of the node and remaining two module requirement is to balance the load between partitions.

3.3 NON FUNCTIONAL REQUIREMENT

1. High Performance
2. Secured
3. Time complexity is managed
4. Better load balancing.
5. Efficiency is provided

3.4 SOFTWARE DESCRIPTIONS

Java Technology

Java technology is both a programming language and a platform.With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

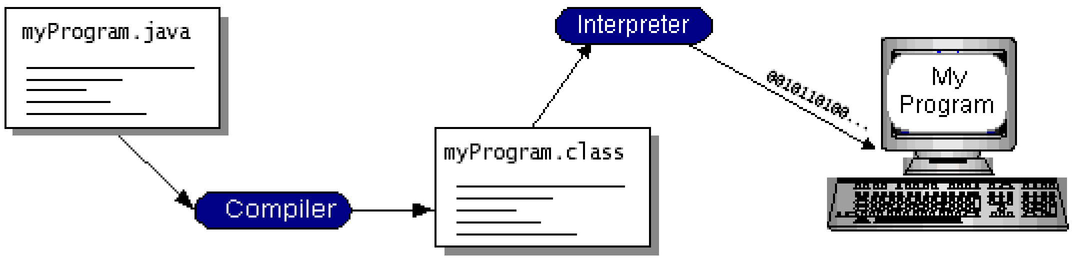


Fig 1: Compilation Process

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

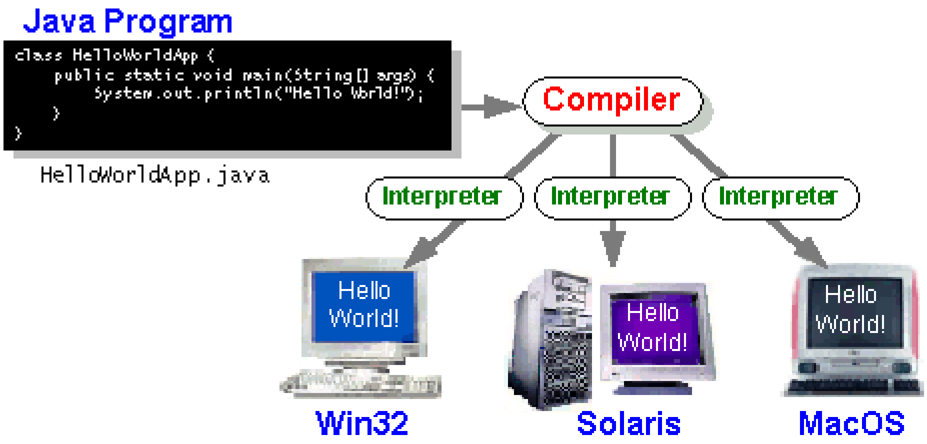


Fig 2 : Java Virtual Machine

The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

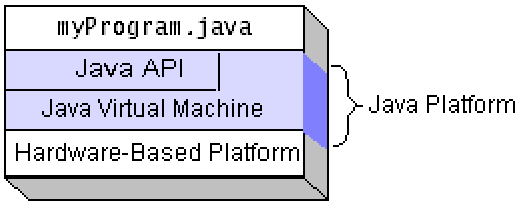
- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java

API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

The Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans ^{TM}, can plug into existing component architectures.
- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Java Database Connectivity (JDBC^{TM}): Provides uniform access to a wide range of relational databases.

Swings

After learning AWT, lets now see what's Swing? Well, Swing is important to develop Java programs with a **graphical user interface (GUI).** There are many components which are used for the building of GUI in Swing. The **Swing Toolkit** consists of many components for the building of GUI. These components are also helpful in providing **interactivity to Java applications.** Following are components which are included in Swing toolkit:

- **list controls**
- **buttons**
- **labels**
- **tree controls**
- **table controls**

All AWT flexible components can be handled by the Java Swing. Swing toolkit contains far more components than the simple component toolkit. It is unique to any other toolkit in the way that it supports integrated internationalization, a highly customizable text package, rich undo support etc. Not only can this have you also created your own look and feel using Swing other than the ones that are supported by it. The customized look and feel can be created using Synth which is specially designed. Not to forget that Swing also contains the basic user interface such as customizable painting, event handling, drag and drop etc.

MY SQL

MySQL is the world's second most widely used open-source relational database management system. It is named after co-founder Michael Widenius's daughter, My. The SQL phrase stands for Structured Query Language.

MySQL, pronounced either "My S-Q-L" or "My Sequel," is an open source relational database management system. It is based on the structure query language (SQL), which is used for adding, removing, and modifying information in the database.

Standard SQL commands, such as ADD, DROP, INSERT, and UPDATE can be used with MySQL.

MySQL can be used for a variety of applications, but is most commonly found on Web servers. A website that uses MySQL may include Web pages that access information from a database. These pages are often referred to as "dynamic," meaning the content of each page is generated from a database as the page loads. Websites that use dynamic Web pages are often referred to as database-driven websites.

Many database-driven websites that use MySQL also use a Web scripting language like PHP to access information from the database. MySQL commands can be incorporated into the PHP code, allowing part or all of a Web page to be generated from database information. Because both MySQL and PHP are both open source (meaning they are free to download and use), the PHP/MySQL combination has become a popular choice for database-driven websites.

SQL yog MySQL GUI

SQLyog is the most powerful MySQL manager and admin tool, combining the features of MySQL Administrator, phpMyAdmin and other MySQL Front Ends and MySQL GUI tools.
SQLyog is a GUI tool for the RDBMS MySQL. It is developed by Webyog, Inc. based out of Bangalore, India and Santa Clara, California.. SQLyog was freely available until v3.0 when it was made commercial software. SQLyog .

3.5 SYSTEM DESIGN

INTRODUCTION

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

1. Detailed design

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

The Below DFD shows the complete data flow in the System.

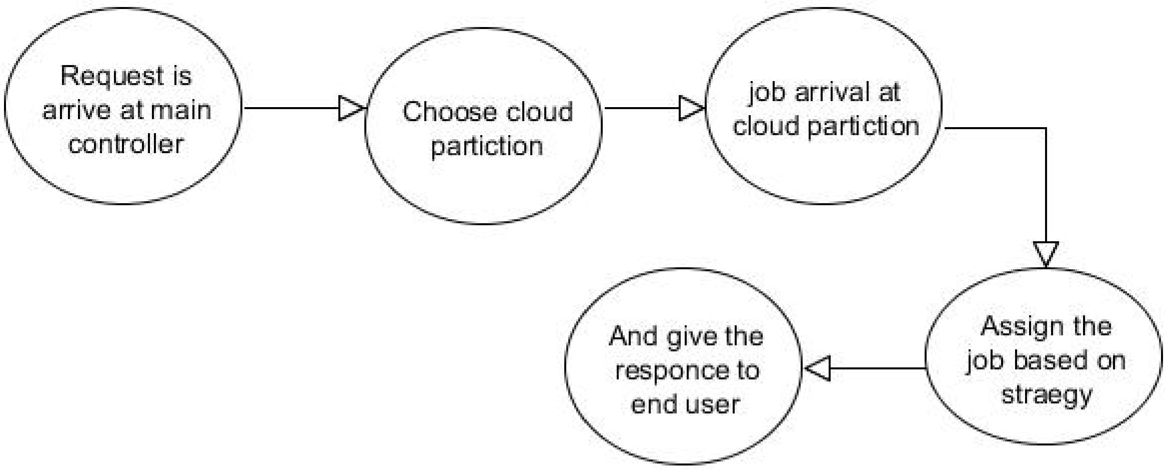


Fig 3 : Data Flow Diagram

3.6 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

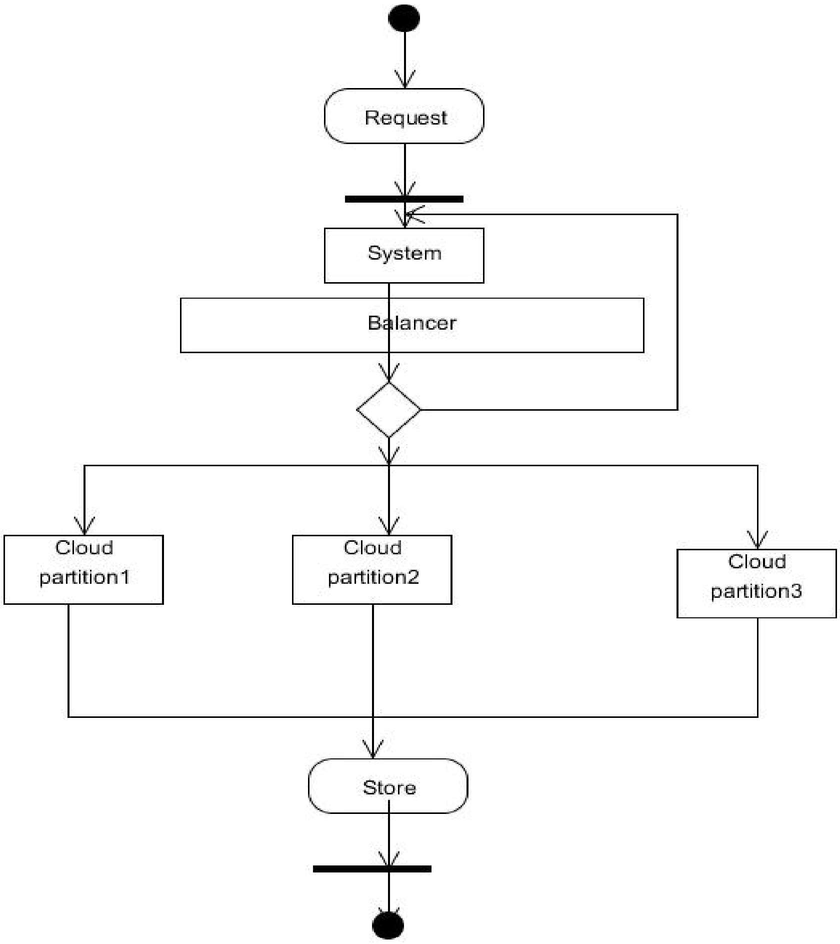


Fig 4 :activity diagram

3.7 Sequence Diagram:

A **sequence diagram** in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically (but not always), are associated with use case realizations in the Logical View of the system under development.

Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner

Below diagram shows the sequence diagram of the proposed system it shows the interaction between different modules of the proposed system.

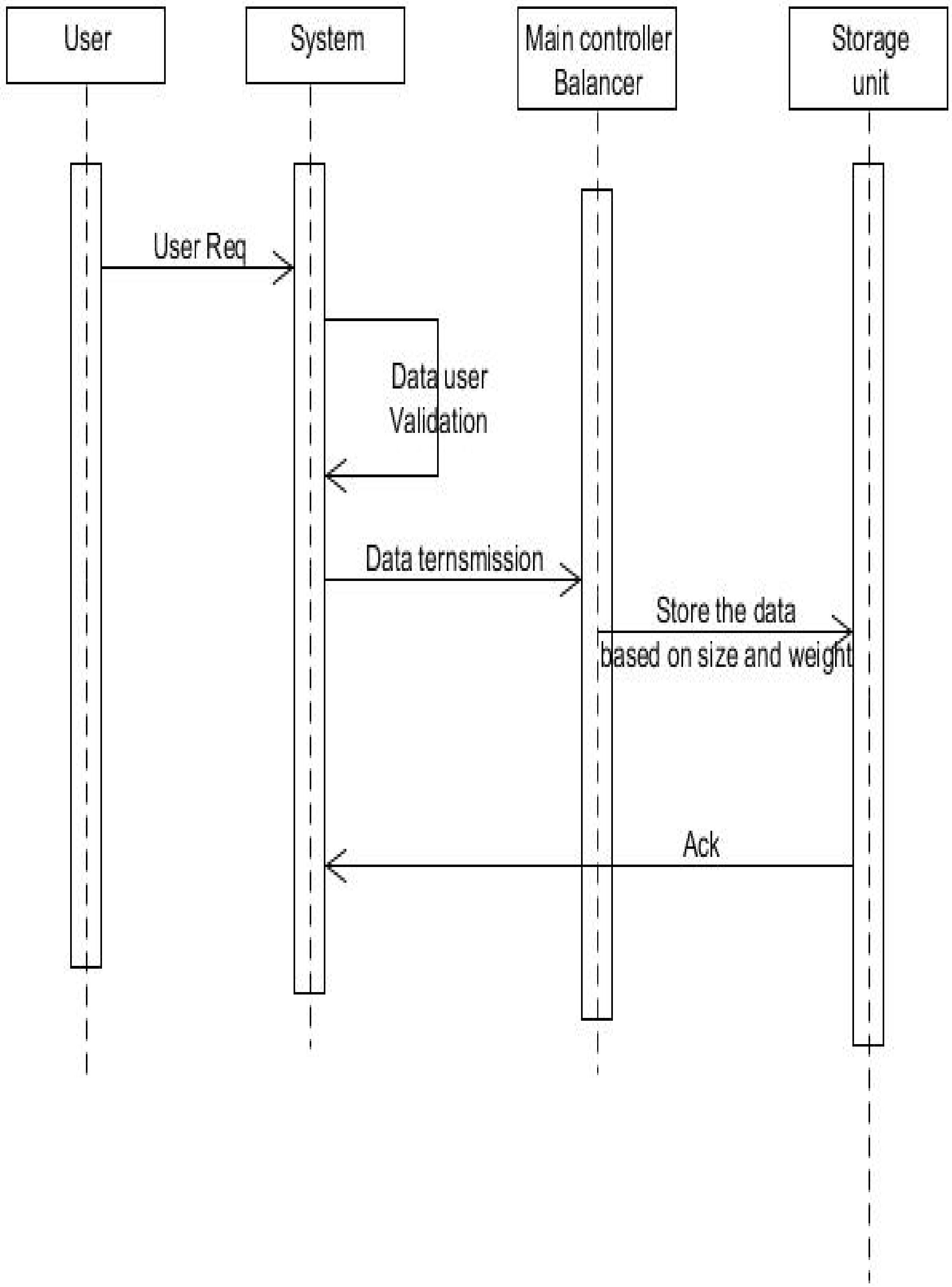


Fig 5 : Sequence diagram

3.8 Use Case Diagram:

A use case in software engineering and systems engineering is a description of a system's behavior as it responds to a request that originates from outside of that system. In other words, a use case describes "who" can do "what" with the system in question. The use case technique is used to capture a system's behavioral requirements by detailing scenario-driven threads through the functional requirements.

The below system shows the user interaction with the system here in this proposed system the user is the system which receive,validate,transfer data .

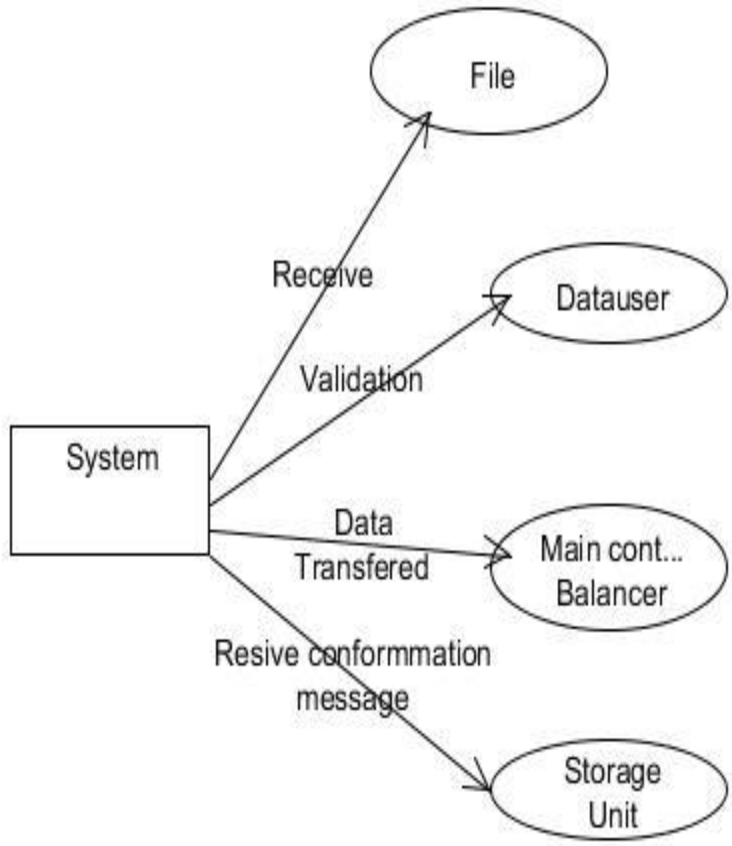


Fig 6 : User system interaction

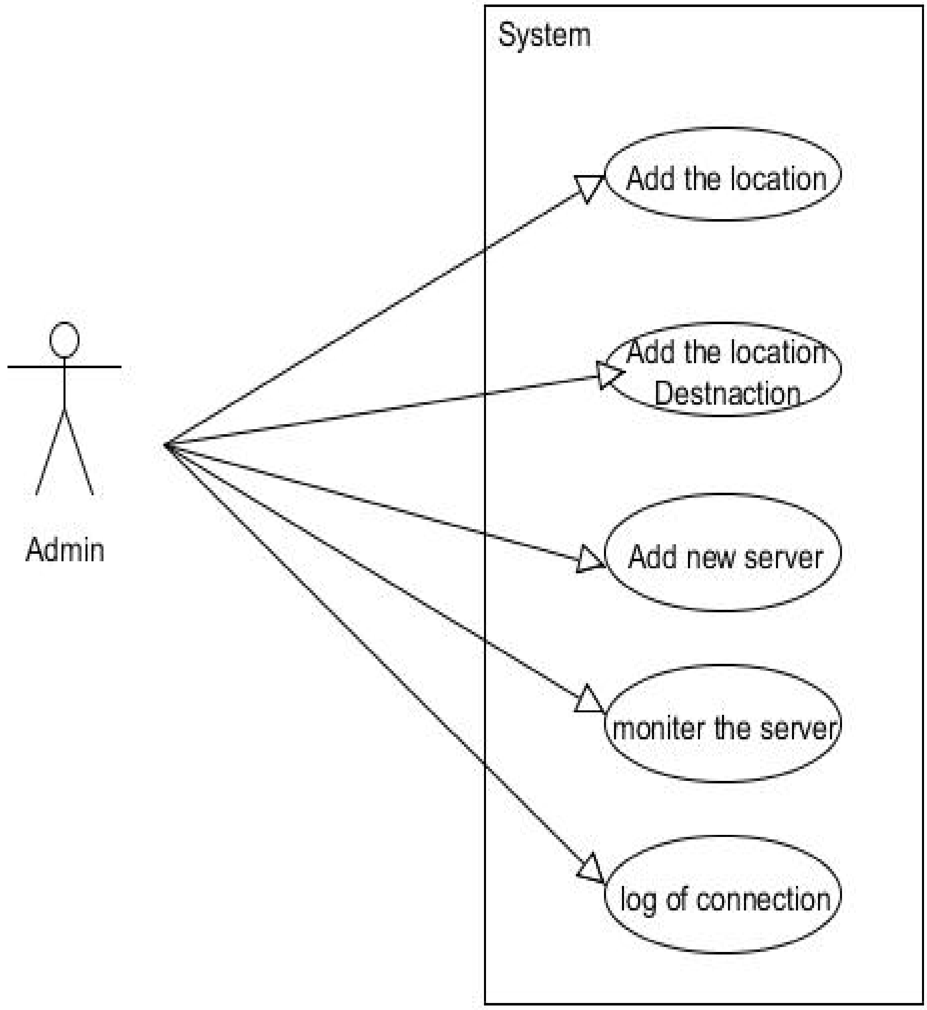


Fig 7 : Use case diagram

3.9 ER DIAGRAM

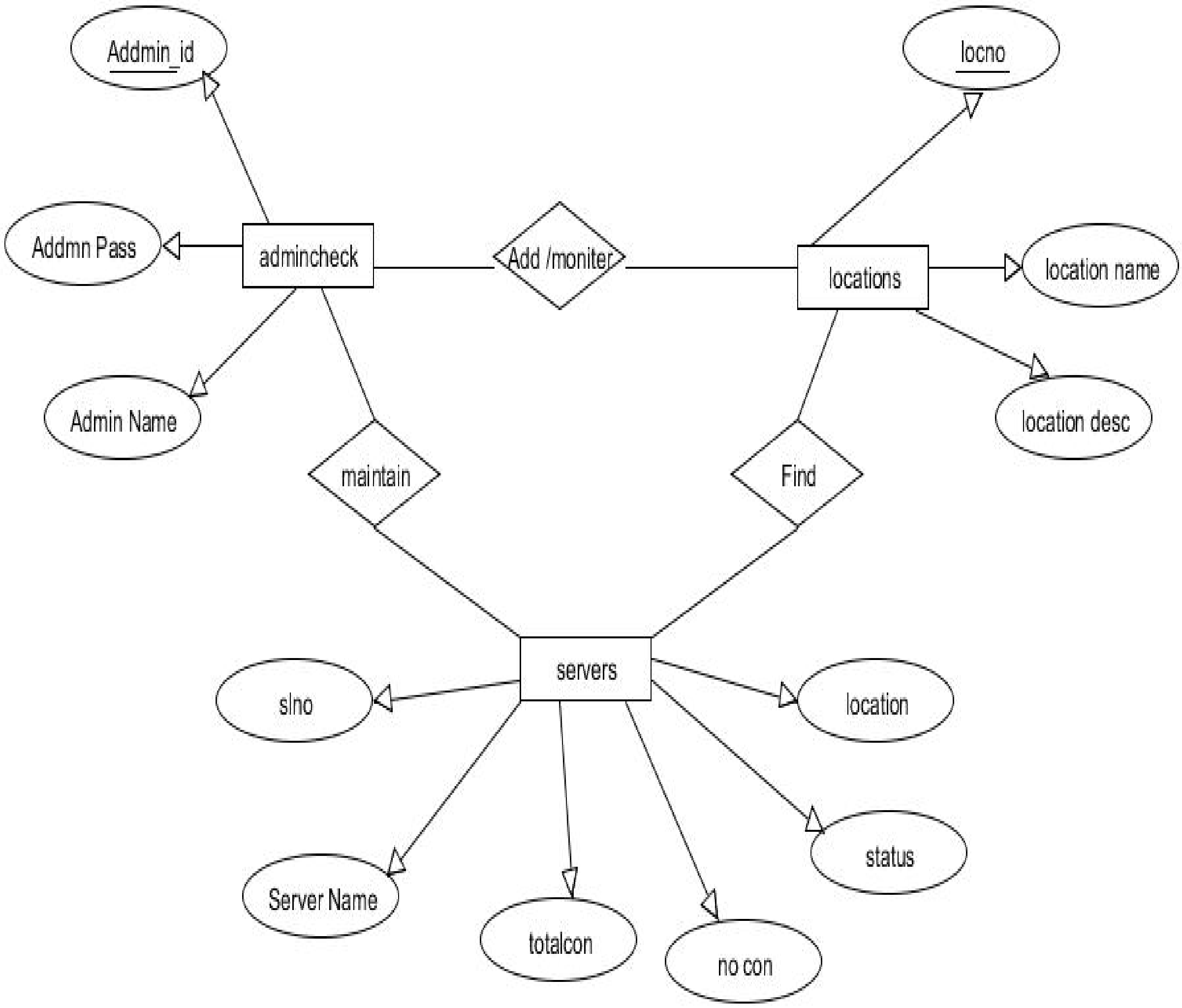


Fig 8 : ER diagram

3.10 Flow chart

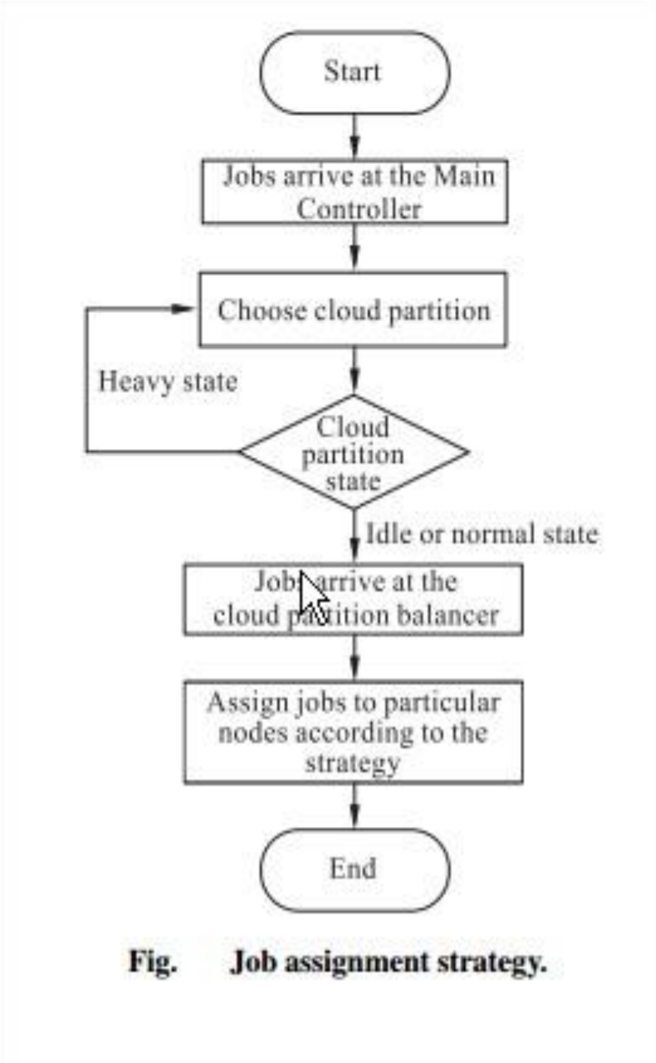


Fig 9 : Flow Chart

Chapter4 : IMPLEMENTATION

4.1 System Model

There are several cloud computing categories with this work focused on a public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider. A large public cloud will include many nodes and the nodes in different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations.

Good load balance will improve the performance of the entire cloud. Better load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. The current model integrates several methods and switches between the load balance methods based on the system status. A relatively simple method can be used for the partition idle state with a more complex method for the normal state. The load balancers then switch methods as the status changes. Here, the idle status uses an improved Round Robin algorithm while the normal status uses a game theory based load balancing strategy.

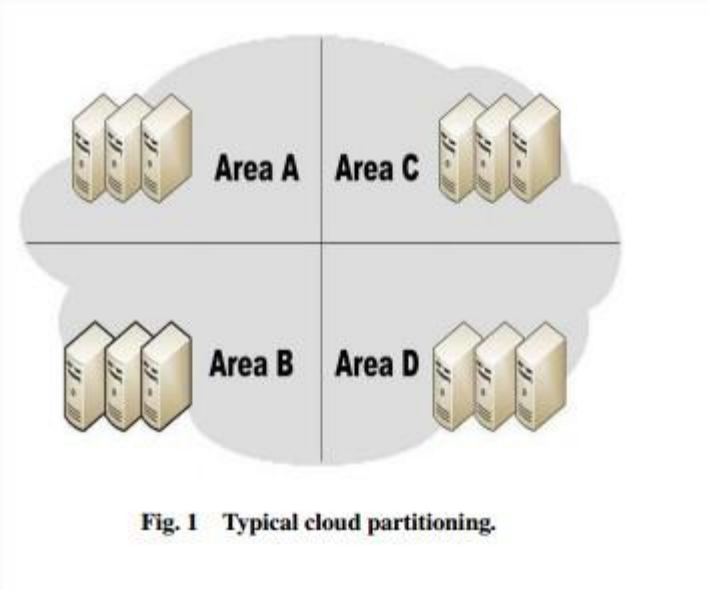


Fig 10 : Partitioning of cloud ref : IEEE TRANSACTIONS ON CLOUD COMPUTING YEAR 2013

4.2 Main Modules:-

1. USER MODULE **:**

In this module, Users are having authentication and security to access the detail which is presented in the ontology system. Before accessing or searching the details user should have the account in that otherwise they should register first.

2. SYSTEM MODEL **:**

There are several cloud computing categories with this work focused on a public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider. A large public cloud will include many nodes and the nodes in different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations. With the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally. If the cloud partition load status is not normal, this job should be transferred to another partition.

3. MAIN CONTROLLER AND BALANCERS:

The load balance solution is done by the main controller and the balancers.
The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals with information for each partition, smaller data sets will lead to the higher processing rates. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs.

4. CLOUD PARTITION LOAD BALANCING STRATEGY:

When the cloud partition is idle, many computing resources are available and relatively few jobs are arriving. In this situation, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method can be used. There are many simple load balance algorithm methods such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin. The Round Robin algorithm is used here for its simplicity.

The load balancing strategy is based on the cloud partitioning concept. After creating the cloud partitions, the load balancing then starts: when a job arrives at the system, with the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally. If the cloud partition load status is not normal, this job should be transferred to another partition. The whole process is shown in Fig.2.

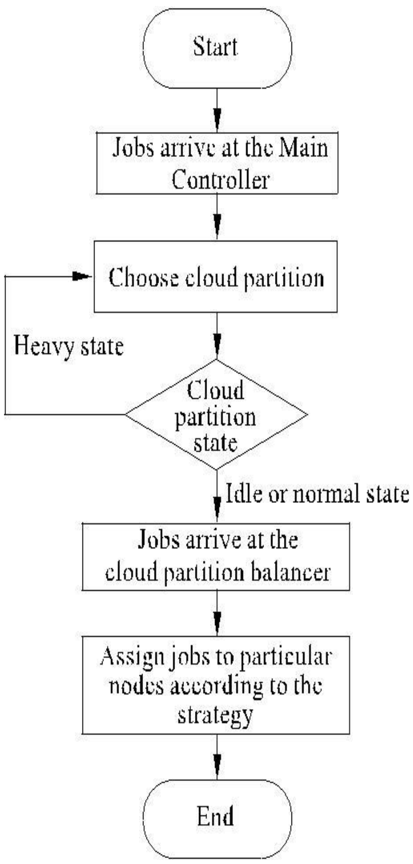


Fig 11: Job assigning .

4.3 Main controller and balancers

The load balance solution is done by the main controller and the balancers.The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals with information for each partition, smaller data sets will lead to the higher processing rates. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs. The relationship between the balancers and the main controller is shown in fig 12;

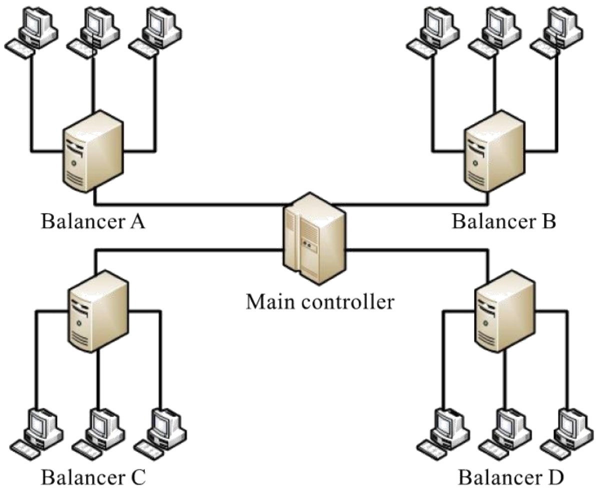


Fig 12 : relation between balancer and main controller

Assigning jobs to the cloud partition

When a job arrives at the public cloud, the first step is to choose the right partition. The cloud partition status can be divided into three types:

(1) Idle: When the percentage of idle nodes exceeds , change to idle status.

(2) Normal: When the percentage of the normal nodes exceeds , change to normal load status.

(3) Overload: When the percentage of the overloaded nodes exceeds , change to overloaded status.

The parameters , , and are set by the cloud partition balancers. The main controller has to communicate with the balancers frequently to refresh the status information. The main controller then dispatches the jobs using the following strategy: When job i arrives at the system, the main controller queries the cloud partition where job is located. If this location's status is idle or normal, the job is handled locally. If not, another cloud partition is found that is not overloaded.

Algorithm 1 Best Partition Searching

```
begin
  while job do searchBestPartition (job);
    if partitionState == idle k partitionState == normal then Send Job to Partition;
    else
      search for another Partition; end if
```

end while end

Assigning jobs to the nodes in the cloud partition

The cloud partition balancer gathers load information from every node to evaluate the cloud partition status. This evaluation of each node's load status is very important. The first task is to define the load degree of each nodes.

The node load degree is related to various static parameters and dynamic parameters. The static parameters include the number of CPU's, the CPU processing speeds, the memory size, etc. Dynamic parameters are the memory utilization ratio, the CPU utilization ratio, the network bandwidth, etc.

The load degree results are input into the Load Status Tables created by the cloud partition balancers. Each balancer has a Load Status Table and refreshes it each fixed period . The table is then used by the balancers to calculate the partition status. Each partition status has a different load balancing solution. When a job arrives at a cloud partition, the balancer assigns the job to the nodes based on its current load strategy. This strategy is changed by the balancers as the cloud partition status changes

4.4 Cloud Partition Load Balancing Strategy

Motivation

Good load balance will improve the performance of the entire cloud. However, there is no common method that can adapt to all possible different situations. Various methods have been developed in improving existing solutions to resolve new problems.

Each particular method has advantage in a particular area but not in all situations. Therefore, the current model integrates several methods and switches between the load balance method based on the system status.

A relatively simple method can be used for the partition idle state with a more complex method for the normal state. The load balancers then switch methods as the status changes. Here, the idle status uses an improved Round Robin algorithm while the normal status uses a game theory based load balancing strategy.

Load balance strategy for the idle status

When the cloud partition is idle, many computing resources are available and relatively few jobs are arriving. In this situation, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method can be used.

There are many simple load balance algorithm methods such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin. The Round Robin algorithm is used here for its simplicity.

The Round Robin algorithm is one of the simplest load balancing algorithms, which passes each new request to the next server in the queue. The algorithm does not record the status of each connection so it has no status information. In the regular Round Robin algorithm, every node has an equal opportunity to be chosen. However, in a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes. Thus, an improved Round Robin algorithm is used , which called “Round Robin based on the load degree evaluation”.

The algorithm is still fairly simple. Before the Round Robin step, the nodes in the load balancing table are ordered based on the load degree from the lowest to the highest. The system builds a circular queue and walks through the queue again and again. Jobs will then be assigned to nodes with low load degrees. The node order will be changed when the balancer refreshes the Load Status Table.
However, there may be read and write inconsistency at the refresh period . When the balance table is refreshed, at this moment, if a job arrives at the cloud partition, it will bring the inconsistent problem. The system status will have changed but the information will still be old. This may lead to an erroneous load strategy choice and an erroneous nodes order. To resolve this problem, two Load Status Tables should be created as: Load Status Table 1 and Load Status Table 2. A flag is also assigned to each table to indicate Read or Write.

When the flag = “Read”, then the Round Robin based on the load degree evaluation algorithm is using this table. When the flag = “Write”, the table is being refreshed, new information is written into this table. Thus, at each moment, one table gives the correct node locations in the queue for the improved Round Robin algorithm, while the other is being prepared with the updated information. Once the data is refreshed, the table flag is changed to “Read” and the other table’s flag is changed to “Write”. The two tables then alternate to solve the inconsistency.

Chapter 5 : CODE IMPLEMENTATION

Administration Implementation

```
package com.vss.sac.dbos;

import com.vss.sac.bussnsonject.LocationsBO;

import com.vss.sac.bussnsonject.NumofConnBO;

import com.vss.sac.bussnsonject.ServersBO;

import com.vss.sac.utill.HibernateUtils;

import java.util.List;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import org.hibernate.Session;

public class AdminImplementation {

 public boolean addlocations(LocationsBO location) {

        Session hbmSession = null;

        boolean STATUS_FLAG = true;

        try {

            hbmSession = HibernateUtils.getSession();

            hbmSession.beginTransaction();

            hbmSession.saveOrUpdate(location);

            hbmSession.getTransaction().commit();

        } catch (Exception ex) {

            // // hbmSession.getTransaction().rollback();

            ex.printStackTrace();

            STATUS_FLAG = false;

        } finally {

            HibernateUtils.closeSession(hbmSession);

        }

        return STATUS_FLAG;    }

public List<LocationsBO> getalllocation() {

        List<LocationsBO> allcat = null;

        Session hbmSession = null;

        boolean STATUS_FLAG = true;
```

```
    try {
        hbmSession = HibernateUtils.getSession();
        hbmSession.beginTransaction();
        allcat = hbmSession.createQuery("from LocationsBO").list();
        hbmSession.getTransaction().commit();
    } catch (HibernateException ex) {
        STATUS_FLAG = false;
    } finally {
        HibernateUtils.closeSession(hbmSession);
    }
    return allcat;
}
public boolean deletelocations(int locid) {

    Session hbmSession = null;
    try {
        hbmSession = HibernateUtils.getSession();
        hbmSession.beginTransaction();
        LocationsBO user = (LocationsBO) hbmSession.get(LocationsBO.class, locid);
        hbmSession.delete(user);
        hbmSession.getTransaction().commit();
    } catch (Exception ex) {
        // // hbmSession.getTransaction().rollback();
        ex.printStackTrace();

    } finally {
        HibernateUtils.closeSession(hbmSession);
    }
    return true;
}
public boolean addservers(ServersBO servers) {
    Session hbmSession = null;
```

```
        boolean STATUS_FLAG = true;
        try {
            hbmSession = HibernateUtils.getSession();
            hbmSession.beginTransaction();
            hbmSession.saveOrUpdate(servers);
            hbmSession.getTransaction().commit();
        } catch (Exception ex) {
            // // hbmSession.getTransaction().rollback();
            ex.printStackTrace();
            STATUS_FLAG = false;
        } finally {
            HibernateUtils.closeSession(hbmSession);
        }
        return STATUS_FLAG;
    }
    public List<ServersBO> getallservers() {
        List<ServersBO> allcat = null;
        Session hbmSession = null;
        boolean STATUS_FLAG = true;
        try {
            hbmSession = HibernateUtils.getSession();
            hbmSession.beginTransaction();
            allcat = hbmSession.createQuery("from ServersBO").list();
            hbmSession.getTransaction().commit();
        } catch (HibernateException ex) {
            STATUS_FLAG = false;
        } finally {
            HibernateUtils.closeSession(hbmSession);
        }
        return allcat;    }
public boolean deleteserver(int srvrid) {
```

```
    Session hbmSession = null;
    try {
        hbmSession = HibernateUtils.getSession();
        hbmSession.beginTransaction();
        ServersBO user = (ServersBO) hbmSession.get(ServersBO.class, srvrid);
        hbmSession.delete(user);
        hbmSession.getTransaction().commit();
    } catch (Exception ex) {
        // // hbmSession.getTransaction().rollback();
        ex.printStackTrace();

    } finally {
        HibernateUtils.closeSession(hbmSession);
    }
    return true;

}

public List<ServersBO> getallserversrandom() {

    List<ServersBO> allcat = null;
    Session hbmSession = null;
    boolean STATUS_FLAG = true;
    try {
        hbmSession = HibernateUtils.getSession();
        hbmSession.beginTransaction();
        allcat = hbmSession.createQuery("from ServersBO WHERE conn < normlcon order by rand()").list();
        hbmSession.getTransaction().commit();
    } catch (HibernateException ex) {
        STATUS_FLAG = false;
    } finally {
        HibernateUtils.closeSession(hbmSession);
```

```
        }

        return allcat;

    }

    public boolean updateconnections(int serverid) {

        List<ServersBO> server = null;
        Session hbmSession = null;
        boolean STATUS_FLAG = true;
        try {
            hbmSession = HibernateUtils.getSession();
            hbmSession.beginTransaction();
            server = hbmSession.createQuery("from ServersBO where serverid=" + serverid + "").list();
            int conncount = server.get(0).getConn();
            conncount = conncount + 1;
            Query query = hbmSession.createQuery("update ServersBO set conn=" + conncount + " where
serverid=" + serverid + "");
            int result = query.executeUpdate();

            hbmSession.getTransaction().commit();
        } catch (Exception ex) {
            // // hbmSession.getTransaction().rollback();
            ex.printStackTrace();
            STATUS_FLAG = false;
        } finally {
            HibernateUtils.closeSession(hbmSession);
        }
        return STATUS_FLAG;
    }
```

```
    public boolean updatenumberconnection(int serverid) {


        NumofConnBO numcon = new NumofConnBO();

        numcon.setServerid(serverid);

        Session hbmSession = null;

        boolean STATUS_FLAG = true;

        try {

            hbmSession = HibernateUtils.getSession();

            hbmSession.beginTransaction();

            hbmSession.saveOrUpdate(numcon);

            hbmSession.getTransaction().commit();

        } catch (Exception ex) {

            // // hbmSession.getTransaction().rollback();

            ex.printStackTrace();

            STATUS_FLAG = false;

        } finally {

            HibernateUtils.closeSession(hbmSession);

        }

        return STATUS_FLAG;


    }


    public List<ServersBO> getallnormlserver() {


        List<ServersBO> allcat = null;

        Session hbmSession = null;

        boolean STATUS_FLAG = true;

        try {

            hbmSession = HibernateUtils.getSession();

            hbmSession.beginTransaction();

            allcat = hbmSession.createQuery("from ServersBO where conn>=normlcon and conn < overcon order by
rand()").list();
```

```
        hbmSession.getTransaction().commit();
    } catch (HibernateException ex) {
        STATUS_FLAG = false;
    } finally {
        HibernateUtils.closeSession(hbmSession);
    }

    return allcat;

}

public List<NumofConnBO> getallactivecon() {

    List<NumofConnBO> allcat = null;
    Session hbmSession = null;
    boolean STATUS_FLAG = true;
    try {
        hbmSession = HibernateUtils.getSession();
        hbmSession.beginTransaction();
        allcat = hbmSession.createQuery("from NumofConnBO").list();
        hbmSession.getTransaction().commit();
    } catch (HibernateException ex) {
        STATUS_FLAG = false;
    } finally {
        HibernateUtils.closeSession(hbmSession);
    }

    return allcat;

}

public boolean deletetheconnandrelesrreso(int delid,int serviceid){
```

```
        List<ServersBO> server = null;

        Session hbmSession = null;

      try {

          hbmSession = HibernateUtils.getSession();

          hbmSession.beginTransaction();

          NumofConnBO user = (NumofConnBO) hbmSession.get(NumofConnBO.class, delid);

          hbmSession.delete(user);

          server = hbmSession.createQuery("from ServersBO where serverid=" + serviceid + "").list();

          int conncount = server.get(0).getConn();

          conncount = conncount - 1;

          Query query = hbmSession.createQuery("update ServersBO set conn=" + conncount + " where
serverid=" + serviceid + "");

          int result = query.executeUpdate();


          hbmSession.getTransaction().commit();

      } catch (Exception ex) {

          // // hbmSession.getTransaction().rollback();

          ex.printStackTrace();


      } finally {

          HibernateUtils.closeSession(hbmSession);

      }

      return true;


   }


}
```

Cloud admin

```
package com.vss.sac.servlets;

import java.io.IOException;
```

```
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name = "CloudAdmin", urlPatterns = {"/CloudAdmin"})
public class CloudAdmin extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
                    String uname=request.getParameter("username");
            String pass=request.getParameter("password");
                    if(uname.equalsIgnoreCase("admin") && pass.equalsIgnoreCase("admin")){
                response.sendRedirect("adminhome.jsp");
            }
            else{
                response.sendRedirect("adminlogin.jsp");
            }
             }
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
```

```
 @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Delete locations:

```
package com.vss.sac.servlets;
import com.vss.sac.dbos.AdminImplementation;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name = "Deleatlocations", urlPatterns = {"/Deleatlocations"})
public class Deleatlocations extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
      int locid=Integer.parseInt(request.getParameter("lid"))
          AdminImplementation adminimp=new AdminImplementation();

            boolean flag=adminimp.deletelocations(locid);
                     if(flag){
               response.sendRedirect("AddLocations.jsp");
          }
        }
    }
```

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Delete server :

```
package com.vss.sac.servlets;
import com.vss.sac.dbos.AdminImplementation;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name = "DeleatServers", urlPatterns = {"/DeleatServers"})
public class DeleatServers extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
```

```
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */


              int srvrid=Integer.parseInt(request.getParameter("serid"));
              AdminImplementation adminimp=new AdminImplementation();
             boolean flag=adminimp.deleteserver(srvrid);
                    if(flag){
              response.sendRedirect("AddServer.jsp");
            }
        }
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Add sever location :

```
package com.vss.sac.servlets;
import com.vss.sac.bussnsonject.ServersBO;
import com.vss.sac.dbos.AdminImplementation;
import java.io.IOException;
import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name = "AddServertoloc", urlPatterns = {"/AddServertoloc"})
public class AddServertoloc extends HttpServlet {
  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
      String servername = request.getParameter("servername");
      String serverip = request.getParameter("serverip");
      int serverloc = Integer.parseInt(request.getParameter("serverloc"));
      int numofcon = Integer.parseInt(request.getParameter("noofcon"));
      int idcon = Integer.parseInt(request.getParameter("idlcon"));
      int nrlcon = Integer.parseInt(request.getParameter("normlcon"));
      int ovcon = Integer.parseInt(request.getParameter("overcon"));
      ServersBO servers=new ServersBO();
      servers.setServername(servername);
      servers.setServerip(serverip);
      servers.setServerloc(serverloc);
      servers.setNumcon(numofcon);
      servers.setIdlcon(idcon);
      servers.setNormlcon(nrlcon);
      servers.setOvercon(ovcon);
      servers.setConn(0);
        AdminImplementation adminimp=new AdminImplementation();

      boolean flag=adminimp.addservers(servers);
      if(flag){
        response.sendRedirect("AddServer.jsp");
```

```
            }
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Chapter 6 : Testing And Result

6.1 SOFTWARE TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS:

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/ Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black *Box* Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Cases

Test case ID	Test case name	Test case description	Test steps				Test status P/F
			Step	I/p given	Expected o/p	Actual o/p	
TC01	Admin login	To verify that the user name and password is correct	Enter user name and password	Un:admin Pw:admin	Admin is ensured with successful login	Admin logs in to the server home page	Pass
	Admin login	To verify user name and password is valid	Enter user name and password	Un:admin Pw:pwadmin	Error message should be displayed	A pop up message is displayed to enter valid user name and password	Fail
TC02	User	Validate the	Enter all the details	Add all the Details of	Accept all the details	Notify successful	Pass

	registration	correctness of the details	of user	the user	and update at the back end	update at the back end	
	User registration	Validate the correctness of the details	If any of the field is left empty	All the details Of the user	Pop up message to enter all the Details	Pop up message to Enter all the details	Fail
TC03	Server Registration	Validate the correctness of the details	Enter all the details of new server	Server details	Accept all the details and update at the back end	Notify successful update at the back end	Pass
	server registration	Validate the correctness of the details	If any of the field is left empty	Server details	Pop up message to enter all the Details	Pop up message to enter all the details	Fail
TC04	Cloud Application	Cloud application Sends request to the server	Cloud request	Cloud request is sent to the Server	Connection is Established from client to the server	Connection is Established from client to the server And cloud services count is incremented	Pass

TC05	Release Connection	Connected Client are released	Release connection	Client connection/ Server connection to the client	Connection to the client from server is released	Connection is released from server to the client And cloud services count is decremented	Pass
							Fail
TC06	Load Balancing	Service request load is distributed among available server	Send the request	Cloud application	Cloud application Request distributed among server Based on round robin and game theory	Cloud application Request distributed among server Based on round robin and game theory	Pass
TC07	Logout	Once the operations are finished Admin comes out of the current page	Select the page to come out	Select logout	Comes out of the current page	Comes out of the page	

Test Results: All the test cases mentioned above passed successfully. No defects encountered

6.2 CONCLUSION

Load balancing in the cloud computing environment has been an important impact on the performance. Good load balancing makes cloud computing more efficient and improves user satisfaction. This article introduced a better load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situation. The algorithm applied the game theory to the load balancing strategy to improve the efficiency in the public cloud environment.

Our proposed cloud clustering technique divides the cloud environment into multiple partitions and simplifies the process load balancing effectively. The algorithm used in this paper is able to automatically supervise the load balancing work through load balancer assigned to each cluster. Thus CPU and Memory can be utilized properly. Thus our proposed technique achieves higher performance, stability, optimal resource utilization, minimize response time and application down time over cloud environment.

6.3 **FUTURE WORK**

Since this work is just a conceptual framework, more work is needed to implement the framework and resolve new problems. Some important points are:

(1) Cloud division rules: Cloud division is not a simple problem. Thus, the framework will need a detailed cloud division methodology. For example, nodes in a cluster may be far from other nodes or there will be some clusters in the same geographic area that are still far apart. The division rule should simply be based on the geographic location (province or state).

(2) How to set the refresh period: In the data statistics analysis, the main controller and the cloud partition balancers need to refresh the information at a fixed period. If the period is too short, the high frequency will influence the system performance. If

the period is too long, the information will be too old to make good decision. Thus, tests and statistical tools are needed to set reasonable refresh periods.

(3) A better load status evaluation: A good algorithm is needed to set Load degree high and Load degree Low , and the evaluation mechanism needs to be more comprehensive.

(4) Find other load balance strategy: Other load balance strategies may provide better results, so tests are needed to compare different strategies. Many tests are needed to guarantee system availability and efficiency.

REFERENCES

1) Gaochao Xu, Junjie Pang, and Xiaodong Fu_, IEEE TRANSACTIONS ON CLOUD COMPUTING, hhttps://wwwgooge/cloudloadbalancing?doccd=986483&ref=jujie,YEAR 2013
2) 1) Z. Chaczko, V. Mahadevan, S. Aslanzadeh, andC. Mcdermid, Availability and load balancing in cloudcomputing, presented at the 2011 International Conferenceon Computer and Software Modeling, Singapore, 2011.
3) 2) K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh,N. Nitin, and R. Rastogi, Load balancing of nodesin cloud using ant colony optimization, in Proc. 14th Interna tional Conference on Computer Modelling andSimulation (UKSim), Cambridgeshire, United Kingdom,Mar. 2012, pp. 28-30.
4) 3) M. Randles, D. Lamb, and A. Taleb-Bendiab, A comparative study into distributed load balancing algorithms for cloud computing, in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications, Perth, Australia, 2010, pp. 551-556.
5) 4) A. Rouse, Public cloud, http://searchcloudcomputing. techtarget.com/defi nition/public-cloud, 2012.
6) 5) D. MacVittie, Intro to load balancing for developers —The algorithms, https://devcentral.f5.c om /blogs/us/introto-load-balancing-for-developers-ndash-the-algorithms,2012.
7) 6) S. Penmatsa and A. T. Chronopoulos, Game-theoreticstatic load balancing for distributed systems, Journal of Parallel and Distributed Computing, vol. 71, no. 4, pp. 537-555, Apr. 2011.
8) 7) D. Grosu, A. T. Chronopoulos, and M. Y. Leung, Load balancing in distributed systems: An approach using cooperative games, in Proc. 16th IEEE Intl. Parallel andDistributed Processing Symp., Florida, USA, Apr. 2002, pp. 52-61.
9) 8) S. Aote and M. U. Kharat, A game-theoretic model for dynamic load balancing in distributed systems, in Proc. The International Conference on Advances in Computing, Communication and Control (ICAC3 ’09), New York, USA, 2009, pp. 235-238