WEB LOAD BALANCING

MAIN PROJECT REPORT

Submitted by

FIROSE MUHAMMED KUTTY T

MANYA MENON

MUHAMMED SHAHEEN P

VYSAKH ANTO

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.TECH)

in

COMPUTER SCIENCE & ENGINEERING

of

UNIVERSITY OF CALICUT

Under the guidance of

ANIL ANTONY

JUNE 2011

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

WEB LOAD BALANCING

MAIN PROJECT REPORT

Submitted by

FIROSE MUHAMMED KUTTY T

MANYA MENON

MUHAMMED SHAHEEN P

VYSAKH ANTO

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.TECH)

in

COMPUTER SCIENCE & ENGINEERING

of

UNIVERSITY OF CALICUT

Under the guidance of

ANIL ANTONY

JUNE 2011

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

Department of Computer Science & Engineering
JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY
THRISSUR 679 531

JUNE 2011

BONAFIDE CERTIFICATE

Certified that this project report **" WEB LOAD BALANCING "** being submitted in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** of **University of Calicut** is the bonafide work of **" FIROSE MUHAMMED KUTTY T,MANYA MENON,MUHAMMED SHAHEEN P,VYSAKH ANTO "**, who carried out the project work under our supervision.

Prof. Muralee Krishnan C
HOD
Dept. of CSE

Anil Antony
PROJECT GUIDE
Asst. Professor
Dept. of CSE

CONTENTS

Acknowledgement **iii**

Abstract **iv**

List of Figures **v**

List of Tables **vi**

1 Introduction **1**

1.1 Overview 1

1.2 Motivation and Technical Relevance 1

1.3 Member roles and responsibilities 2

1.4 Layout 2

2 Literature Survey **4**

2.1 Types Of Load Balancing 5

2.1.1 Hardware Load Balancing 5

2.1.2 Software Load Balancing 10

2.2 Routing and Load Balancing 14

2.3 Load Balancer Features 15

2.4 Load Balancing In Telecommunications 17

2.5 Relationship With Failover 17

2.6 Persistence 17

2.7 WAPT 19

3 Proposed System **20**

3.1 Development and plan 20

3.1.1 The Incremental Model 20

3.1.2 The Spiral Model 21

4 System Requirements Specification **23**

4.1 Software Requirements 23

4.2 Hardware Requirements 23

5 Design & Analysis **24**

5.1 System Analysis 24
5.1.1 Module breakup 24
5.1.2 Member effort 24

5.2 System Design 25
5.2.1 Use Case Models / Flow Diagrams 25

6 Implementation **26**

6.1 Introduction 26
6.1.1 User Module 26
6.1.2 Web load balancer Module 26
6.1.3 Log details Module 26
6.1.4 Server Module 26
6.1.5 Application Module 26

6.2 Screenshots 27

6.3 Pseudo codes 30

7 Testing & Maintenance **46**

7.1 Tests 46
7.1.1 Unit Testing 46
7.1.2 Integration Testing 47
7.1.3 Validation Testing 47
7.1.4 User Acceptance Testing 47

7.2 Maintenance 47

8 Conclusion **49**

8.1 Introduction 49

8.2 Future work 49

References **50**

ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to all respected personalities who had guided, inspired and helped us in the successful completion of this project.

First and foremost, we express our thanks to **The Lord Almighty** for guiding us in this endeavour and making it a success.

We are thankful to our Principal **Dr. U Lazar John** and the Management for providing us with excellent lab and infrastructure facilities.

Our sincere thanks to the Head of the Department of Computer Science & Engineering, Prof. **Muralee Krishnan C** for his valuable guidance and suggestions.

We would like to express our deepest gratitude to **Mr. Anil Antony** for his valuable contributions and guidance.

Last but not least, we thank all our teaching and non teaching staffs of Department of Computer Science & Engineering, and also our friends for their immense support and help in all the stages for the development of the project.

ABSTRACT

A server is limited in how many users it can serve in a given period of time, and once it hits that limit the only options are to replace it with a newer, faster machine, or add another server and share the load between them. A load balancer can distribute connections among two or more servers, proportionally cutting the work each has to do. Our project is to balance the load of servers and distribute the load among a certain number of servers. So far, user interface at the user(client) have been completed. Now coding of web load balancer is at the point of completion.Load balancing is extremely important and it is fundamental to the operational success of some of the most recognised high traffic websites visited today.

List of Figures

2.1 Basic Load Balancer 5
2.2 Direct routing load balancing method 6
2.3 NAT load balancing method 7
2.4 SNAT load balancing method 8
2.5 SNAT-TPROXY load balancing method 9
2.6 SSL Termination or Acceleration (SSL) with or without TPROXY 10
2.7 DNS load balancing 11
2.8 Reverse proxying 12

5.1 Usecase model for web load balancer 25

6.1 Screenshot 1-user 27
6.2 Screenshot 2-log details 27
6.3 Screenshot 3-server process 28
6.4 Screenshot 4-controller 28
6.5 Screenshot 3-server process 29
6.6 Screenshot 4-controller 29

List of Tables

1.1 Team Organization 2

5.1 Module Description 24
5.2 Module Allocation 24

7.1 Unit test chart 46

CHAPTER 1
Introduction

1.1 Overview

Over the last decade-and-a-half, Internet use has dramatically increased, placing an extraordinarily high level of demand on underlying hardware.In order to keep up with the increase in user requests and preclude saturation of hardware resources, the hardware itself has become much more powerful and capable.However, the key to successfully serving a customer base that continues to grow is recognition that the solution cannot be achieved merely by investing large sums of money in the latest and greatest hardware.

Rather, the answer lies in an understanding of how the network can be used to your advantage and how you can distribute requests to many servers within a cluster that can then process them in an expeditious manner.This concept, is called load balancing and appropriately used, it can help ensure that no server becomes so overburdened with requests that it ends up failing to properly function.

1.2 Motivation and Technical Relevance

A Web server (program) has defined load limits, because it can handle only a limited number of concurrent client connections (usually between 2 and 80,000, by default between 500 and 1,000) per IP address (and TCP port) and it can serve only a certain maximum number of requests per second depending on its own settings, the HTTP request type, the hardware and software limits of the OS where it is working etc. As a result our team got inspired to develop a project so as to reduce the load of the server and make application access much more faster.

In networking, load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy. The load balancing service is usually provided by a dedicated program or hardware device (such as a multilayer switch or a DNS server).Load balancing is extremely

important and it is fundamental to the operational success of some of the most recognized, high-traffic Websites visited today.Hotmail, MSN, Yahoo, Google, CNN, and USATODAY all have content that they are providing to millions of Internet users. In order to effectively deliver this content, these Websites have been deployed on a large number of servers that are clustered together.[1]

1.3 Member roles and responsibilities

After a brief overview of team organization, a table showing the team roles and responsibilities can be shown as below. Since Shaheen had good leadership quality,he was made the

1.1: Team Organization

Name	**Responsibility**
Muhammed Shaheen P	Leader
Manya Menon	Designer
Firose Muhammed Kutty T	Debugger
Vysakh Anto	Programmer

leader of our group. Designing skills of Manya is good,so she was given the designing responsibility. Vysakh is good at programming so he was given programming responsibility. Firose is a good debugger, hence he was given the debugging responsibility.

1.4 Layout

Here is a brief outline of the contents of the chapters to follow.

Chapter 2 presents the relevant documents referenced during the initial survey of the project concept.

Chapter 3 presents the process model and relevant details which suits our project.

Chapter 4 includes the hardware and software requirements for the project.

Chapter 5 gives an overview of the schedule of the Term project work. Includes member work effort and module allocations to each member as per his/her responsibility.The section also presents the general architecture of our project concept.

Chapter 6 includes the program code elements for the initial model (working implementation) of the project.

Chapter 7 includes the details of the unit tests, integration tests and proposals for future maintenance.

The last chapter, Chapter 8 summarizes the work done in our project.

CHAPTER 2

Literature Survey

According to our literature survey, load balancing is an already implemented concept. In computer networking, load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Load balancing is extremely important and it is fundamental to the operational success of some of the most recognized, high-traffic Websites visited today. Hotmail, MSN, Yahoo, Google, CNN, and USATODAY all have content that they are providing to millions of Internet users. In order to effectively deliver this content, these Websites have been deployed on a large number of servers that are clustered together.

In order to determine how load balancing can benefit your organization, you will need to take some time to perform capacity planning. Evaluate how large your current customer base is, project growth rates, and use this research to build a configuration that can effectively respond to requests. And remember: load balancing, while primarily applied to web servers, can be used with a variety of other services.

Over the last decade-and-a-half, Internet use has dramatically increased, placing an extraordinarily high level of demand on underlying hardware. In order to keep up with the increase in user requests and preclude saturation of hardware resources, the hardware itself has become much more powerful and capable. However, the key to successfully serving a customer base that continues to grow is recognition that the solution cannot be achieved merely by investing large sums of money in the latest and greatest hardware. Rather, the answer lies in an understanding of how the network can be used to your advantage and how you can distribute requests to many servers within a cluster that can then process them in an expeditious manner. This concept, is called load balancing and appropriately used, it can help ensure that no server becomes so overburdened with requests that it ends up failing to properly function. Load balancing has been around for years. Some load balancing implementations have been hardware-based while others only required installation of special software.[2]

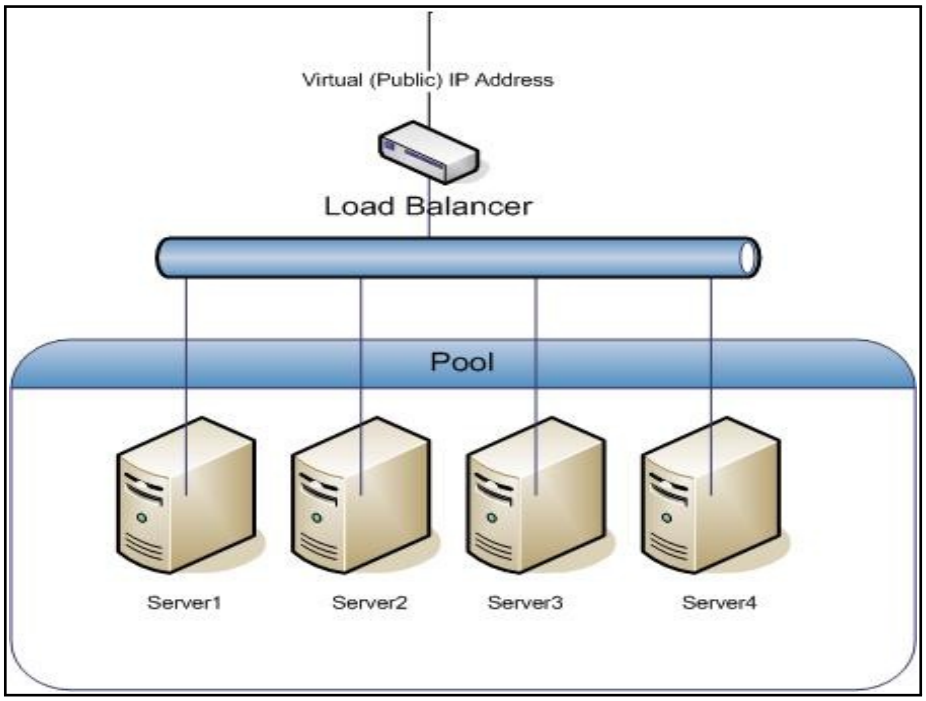

Fig. 2.1: Basic Load Balancer

2.1 Types Of Load Balancing

Load balancing can be hardware or software[3]:

2.1.1 Hardware Load Balancing

Hardware load balancers can route TCP/IP packets to various servers in a cluster. These types of load balancers are often found to provide a robust topology with high availability, but comes for a much higher cost.

Direct Routing (DR) load balancing method

The one-arm direct routing (DR) mode is the recommended mode for Loadbalancer.org installation because it's a very high performance solution with very little change to your existing infrastructure. NB. Foundry networks call this Direct Server Return and F5 call it N-Path.

Direct routing works by changing the destination MAC address of the incoming packet on the fly which is very fast. It means that when the packet reaches the real server it expects it

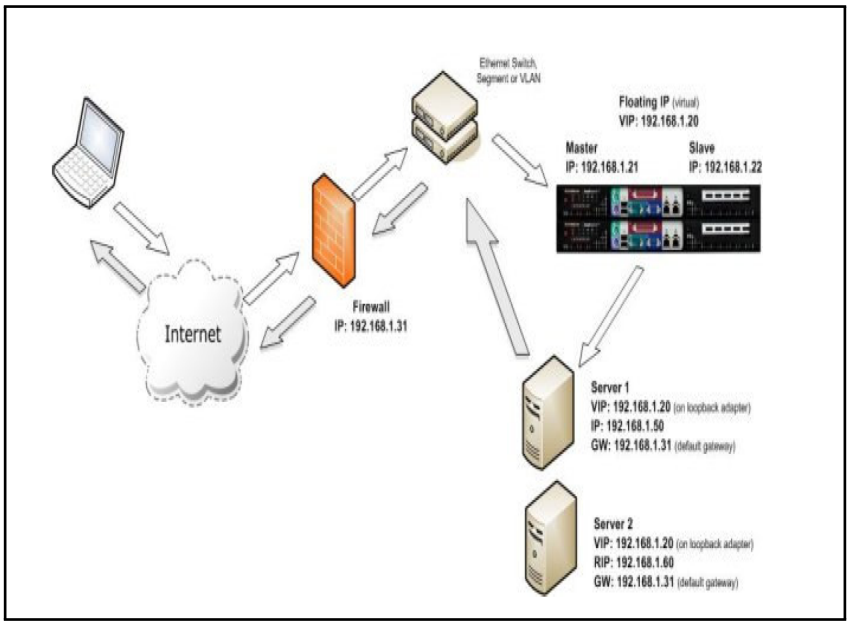

Fig. 2.2: Direct routing load balancing method

to own the VIP.This means you need to make sure the real server responds to the VIP, but does not respond to ARP requests.On average,DR mode is 8 times quicker than NAT for HTTP,50 times quicker for terminal services and much, much faster for streaming media or FTP. Direct routing mode enables servers on a connected network to access either the VIPs or RIPs.No extra subnets or routes are required on the network. The real server must be configured to respond to both the VIP and its own IP address.Port translation is not possible in DR mode i.e. have a different RIP port than the VIP port.

When using a load balancer in one-arm DR mode all load balanced services can be configured on the same subnet as the real servers. The real servers must be configured to respond to the virtual server IP address as well as their own IP address.

Network Address Translation (NAT) load balancing method

Sometimes it is not possible to use DR mode. The two most common reasons being: if the application cannot bind to RIP and VIP at the same time; or if the host operating system cannot be modified to handle the ARP issue. The second choice is Network Address Translation (NAT) mode. This is also a fairly high performance solution but it requires the implementation of a two arm infrastructure with an internal and external subnet to carry out the translation (the same way a firewall works). Network engineers with experience of hardware load balancers will have often used this method.

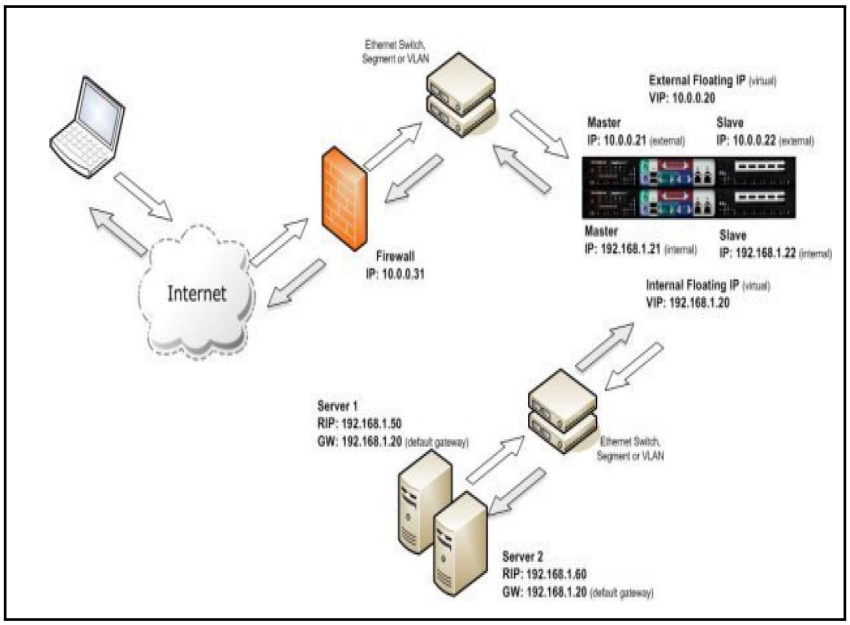

Fig. 2.3: NAT load balancing method

In two-arm NAT mode the load balancer translates all requests from the external virtual server to the internal real servers.The real servers must have their default gateway configured to point at the load balancer.For the real servers to be able to access the internet on their own, i.e. browse the web, the setup wizard automatically adds the required MASQUERADE rule in the firewall script (some vendors incorrectly call this S-NAT).If you want real servers to be accessible on their own IP address for non-load balanced services, i.e. SMTP, you will need to set up individual SNAT and DNAT firewall script rules for each real server. Or you can set up a dedicated virtual server with just one real server as the target.

When using a load balancer in two-arm NAT mode, all load balanced services can be configured on the external IP. The real servers must also have their default gateways directed to the internal IP. You can also configure the load balancers in one-arm NAT mode, but in order to make the servers accessible from the local network you need to change some routing information on the real servers.

It is possible to add routing rules to the real servers in order to perform NAT load balancing on a single subnet(1 arm).

Source Network Address Translation(SNAT) load balancing method

If your application requires that the load balancer handles cookie insertion then you need to use the SNAT configuration. This also has the advantage of a one arm configuration and does

not require any changes to the application servers. However, as the load balancer is acting as a full proxy it doesn't have the same raw throughput as the routing based methods.

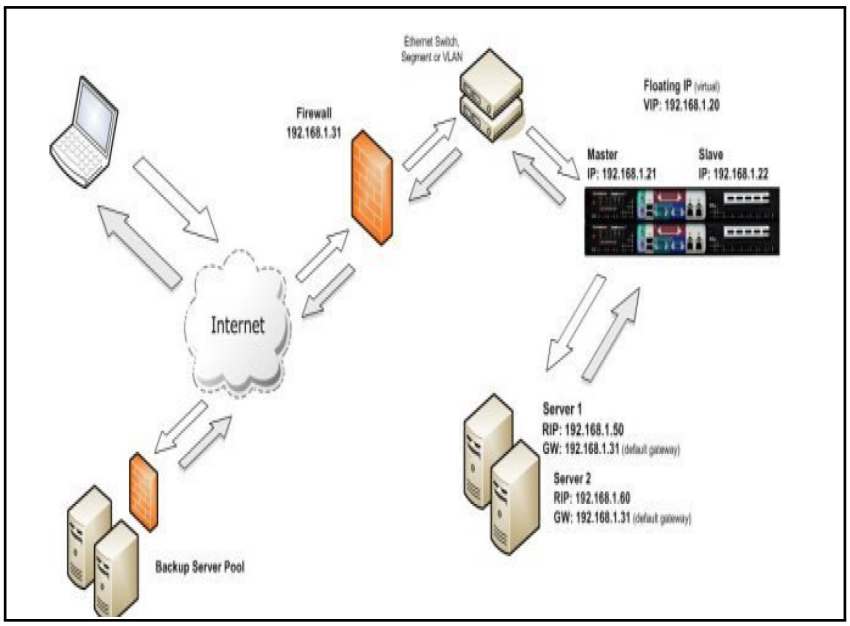

Fig. 2.4: SNAT load balancing method

The network diagram for the Layer 7 HAProxy SNAT mode is very similar to the Direct Routing example except that no re-configuration of the real servers is required. The load balancer proxies the application traffic to the servers so that the source of all traffic becomes the load balancer.

As with other modes a single unit does not require a Floating IP.SNAT is a full proxy and therefore load balanced servers do not need to be changed in any way.

Because SNAT is a full proxy any server in the cluster can be on any accessible subnet including across the Internet or WAN.SNAT is not TRANSPARENT by default i.e. the real servers will see the source address of each request as the load balancers IP address. The clients source IP address will be in the x-forwaded for header (see TPROXY method).

Transparent Source Network Address Translation (SNAT-TPROXY) load balancing method

If the source address of the client is a requirement then HaProxy can be forced into transparent mode using TPROXY, this requires that the real servers use the load balancer as the default gateway (as in NAT mode) and only works for directly attached subnets (as in NAT mode).

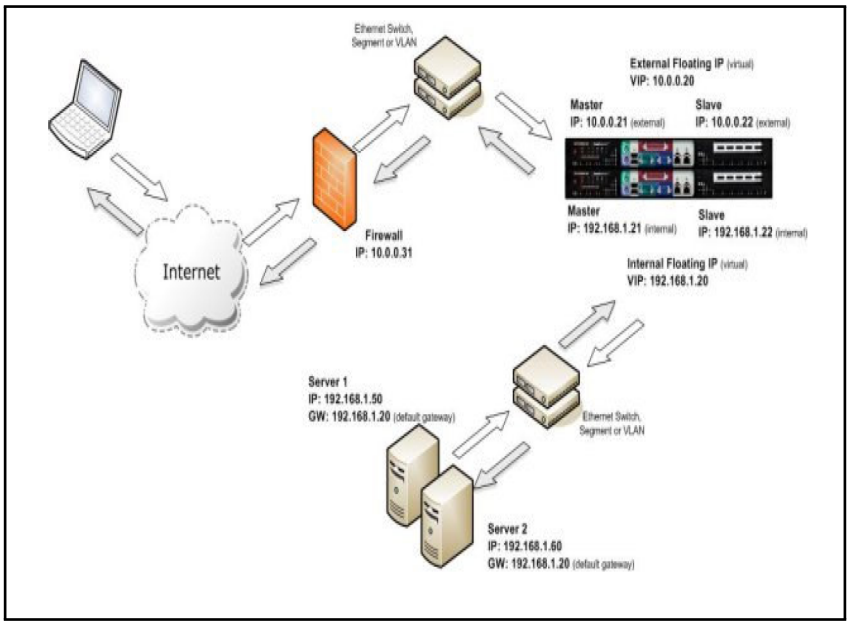

Fig. 2.5: SNAT-TPROXY load balancing method

As with other modes a single unit does not require a Floating IP.SNAT acts as a full proxy but in TPROXY mode all server traffic must pass through the load balancer.The real servers must have their default gateway configured to point at the load balancer.

Transparent proxy is impossible to implement over a routed network i.e. wide area network such as the Internet. To get transparent load balancing over the WAN you can use the TUN load balancing method (Direct Routing over secure tunnel) with Linux or UNIX based systems only.

SSL Termination or Acceleration (SSL) with or without TPROXY

All of the layer 4 and Layer 7 load balancing methods can handle SSL traffic in passs through mode i.e. the backend servers do the decryption and encryption of the traffic. This is very scaleable as you can just add more servers to the cluster to gain higher Transactions per second (TPS). However if you want to inspect HTTPS traffic in order to read or insert cookies you will need to decode (terminate) the SSL traffic on the load balancer. You can do this by imoprting your secure key and signed certificate to the load balancer giving it the authority to decrypt traffic. The load balancer uses standard apache/PEM format certificates.

You can define a Pound SSL virtual server with a single backend either a Layer 4 NAT mode virtual server or more usually a Layer 7 HAProxy VIP which can then insert cookies.

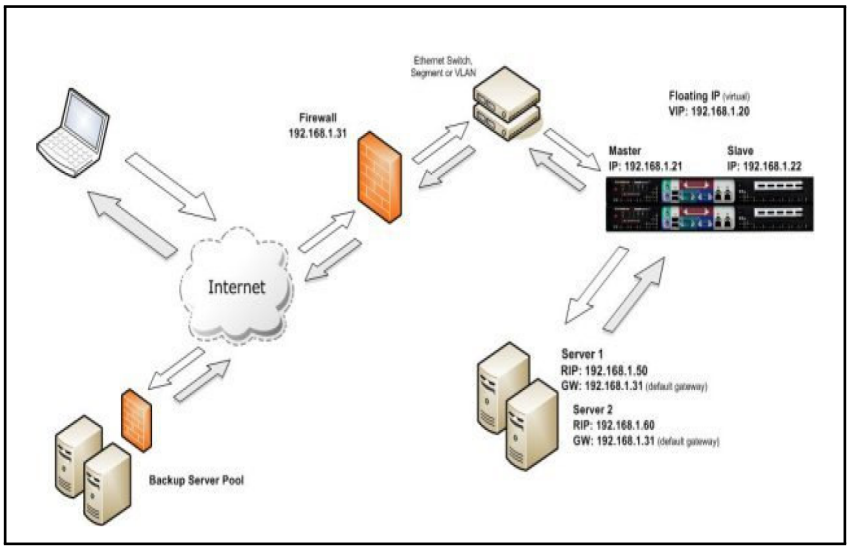

Fig. 2.6: SSL Termination or Acceleration (SSL) with or without TPROXY

Pound-SSL is not TRANSPARENT by default i.e. the backen will see the source address of each request as the load balancers IP address. The clients source IP address will be in the x-forwaded for header. However Pound-SSL can also be configured with TPROXY to ensure that the backend can see the source IP address of all traffic.

2.1.2 Software Load Balancing

These can use different methods. round robin, least heavily loaded, most heavily loaded,etc.Like everything else in life, your Web server too has a constraint on the number of pages it can serve simultaneously to clients (usually browsers). You will especially appreciate this if you serve dynamic content generated in response to user inputCGI scripts, for exampleas this depends heavily on database access and use of other server-side resources; or run a website whose popularity has outgrown its current means of serving.

One way of extending this limit is load balancing.Web servers serve pages to clients as and when a request is made. Whenever a server receives a request, it creates a child process, which handles that particular request. As a result, most Web servers run in multi-threading and multi-processing environments. However, even such an environment puts a limit on the number of Web pages that can be served concurrently, largely because of two factors: the bandwidth available and the Web server itself. Assuming that you have sufficient bandwidth, the performance of your Web server becomes the critical factor.

Your Web servers performance is determined to a large extent by the underlying hardware resources available to it. This limit is higher when the content delivered is static like images

or text, but considerably lower when dealing with dynamic content. Load balancing involves spreading the load among multiple machines, or sometimes even among multiple sites, thereby increasing the resources available. Load balancing in its crudest form would, for example, involve placing all HTML files on one host, all images on another and all CGI scripts on the third. Real-life load balancing, however, involves carefully examining access patterns of various files on the website and keeping identical copies of the same Web server and distributing the load amongst them.

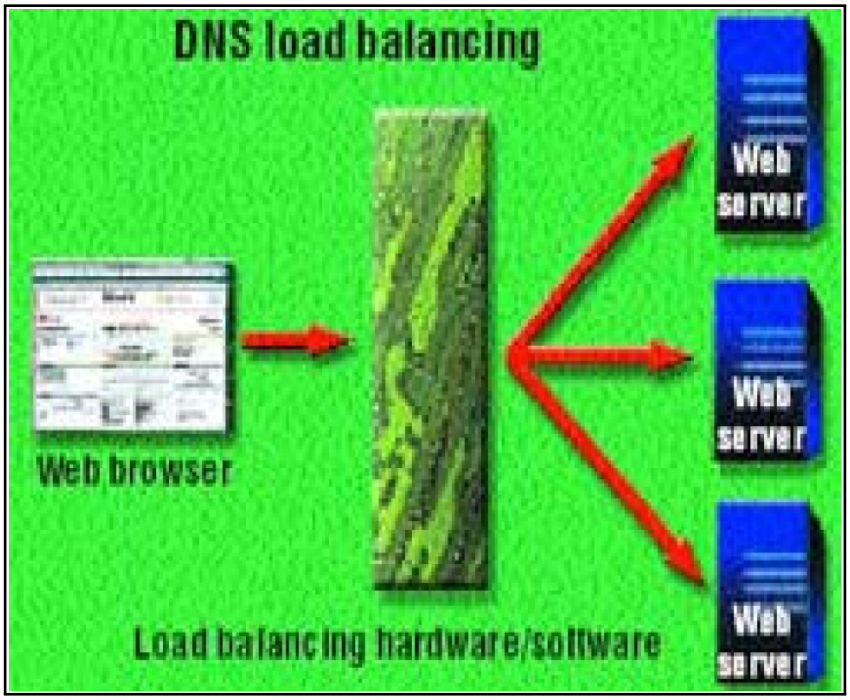

Fig. 2.7: DNS load balancing

One technique, called DNS load balancing, involves maintaining identical copies of the site on physically separate servers. The DNS entry for the site is then set to return multiple IP addresses, each corresponding to the different copies of the site. The DNS server then returns a different IP address for each request it receives, cycling through the multiple IP addresses. This method gives you a very basic implementation of load balancing. However, since DNS entries are cached by clients and other DNS servers, a client continues to use the same copy during a session. This can be a serious drawback, as heavy website users may get the particular IP address that is cached on their client or DNS server, while less-frequent users get another. So, heavy users could experience a performance slowdown, even though the servers resources may be available in abundance.

Another load-balancing technique involves mapping the site name to a single IP address, which belongs to a machine that is set up to intercept HTTP requests and distribute them among multiple copies of the Web server. This can be done using both hardware and software. hard-

ware solutions, even though expensive, are preferred for their stability. This method is preferred over the DNS approach, as better load balancing can be achieved. Also, these load balancers can see if a particular machine is down, and accordingly divert the traffic to another address dynamically. This is in contrast to the DNS method, where a client is stuck with the address of the dead machine, until it can request a new one.

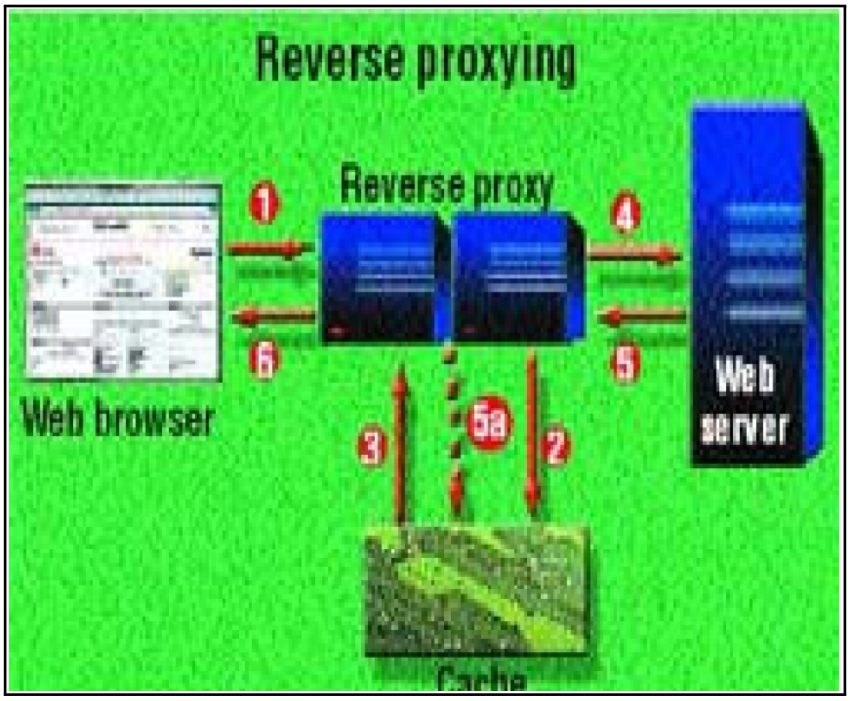

Fig. 2.8: Reverse proxying

Another technique, reverse proxying, involves setting up a reverse proxy, that receives requests from the clients, proxies them to the Web server and caches the response onto itself on its way back to the client. This means that the proxy server can provide static content from its cache itself, when the request is repeated. This in turn ensures that the server itself can focus its energies on delivering dynamic content. Dynamic content cannot generally be cached, as it is generated real time. Reverse proxying can be used in conjunction with the simple load-balancing techniques discussed earlierstatic and dynamic contents can be split across different servers and reverse proxying used for the static content Web server only.

Round Robin DNS was used to manage server congestion. With Round Robin, a DNS server contains multiple A records for a single host, e.g., the Internet resource www.auerbach-publications.com might correspond to three Internet Protocol (IP) addresses: 208.254.79.10, .11, and .12. The machines with these IP addresses are all identically configured each is running a web server that has a complete copy of the Auerbach Publications Website, so no matter which server a request is directed to, the same response is provided. This elementary load balancing mechanism works as soon as a DNS query is made. When a client attempts to

access the Website, a local DNS lookup is performed to determine what the corresponding IP address is. The first time this query is made, the remote DNS server returns all the address records it has. The local DNS server then determines what address record to return to the client. If all records are returned, the client will take the first one that it is given. With each request, the Round Robin algorithm rotates the order in which the address records are returned, so each DNS query will result in a client using a different IP address. When the fourth query is made, the address records are returned in the same order as the first. This process effectively distributes the load across all servers. However, there are a number of significant disadvantages in using it. It sends a client that connects repeatedly to different servers. Doing so breaks applications that maintain state between connections in the server, including most modern web applications.

Load balancing allows Web servers to be scaleable, that is, a server can be scaled up or scaled down. For example if the load on your website is balanced across three identical servers and it experiences a sudden flood of users beyond its current capacity, you can set up additional servers identical to the ones already running and modify the DNS entry of your website to include pointers to these additional hosts. This is called scaling up. Conversely, if you know that your website is particularly low on traffic during certain times, on Sundays, for example, you can remove one or more servers and put their resources to better use. This is scaling down. This involves prior knowledge of access patterns and a little bit of foresight.

These techniques apart, it is important to have the servers basics right toofast CPUs, plenty of RAM, faster (and fatter) disks, etc. Check to ensure that your operating system supports symmetric multiprocessing (SMP), else having multiple processors is a waste. Turning off reverse DNS (looking up the names of addresses accessing your site), if not needed, also provides a considerable increase in performance.

Ideally you should install a load balancer on a dedicated machine that can handle all the incoming connections, with a separate network interface for internal and external connections .Just as servers can be pulled from a cluster when problems arise, they can also be added.

Acting as a virtual server, ”LB(LOAD BALANCER)” transparently routes and load balances incoming requests to a pool of HTTP servers. It can route requests based on the content type in the URL using a load-balancing algorithm.

2.2 Routing and Load Balancing

Acting as a virtual server, LB transparently routes and load balances incoming requests to a pool of servers as follows:

- LB listens for requests on the HTTP, HTTPS, and BHTTP ports.
- The load balancer determines the set of servers capable of handling the request. It does this by building a list of servers that are capable of handling the requested content type.
- Using the configured algorithm, the LB selects a server to process the user request.
- After the initial request, all subsequent requests during that clients sessions are routed to the same server.
- If the server to which a request was to be routed has gone down or goes down while processing a request, LB will return an error page to the browser.

We can add new servers to this pool, remove a server from service, or restart a server. Add a server to the pool of load balancing servers:-

- Take servers in the load balancing pool out-of-service without any interruption in service to the users of the site.
- Route requests based on the content type in the URL using the least connections, round robin, or average response time algorithm.
- Prioritize requests based on the URL-level.
- Retry requests on a new server, if the server to which the request was originally routed to has gone down or goes down during the processing of the request.
- Create customized error messages that provide users with more information, enable users to remedy their current situation, or prompt users to notify you of a problem.

LB can be configured to use one of the following three different load balancing algorithms for determining the server to which an initial request is sent:

- Round Robin-The round robin load balancing algorithm uses a next server marker to identify the next server to which the load balancer should send a request. Whenever a server is selected, the next server marker moves to the next server on the server list or loops back to the beginning when it reaches the end of the list.
- Least Connections-The least connections load balancing algorithm determines which

server has the least number of connections open to it and sends the next request to that server. For each new request LB receives, the least connections algorithm determines the server with the least connections.

- Average Response Time-The average response time load balancing algorithm maintains information on the length of time each server takes to respond to requests. When an new initial request must be routed, it chooses the server that has, historically, responded the fastest.

2.3 Load Balancer Features

Hardware and software load balancers can come with a variety of special features.[4]

- Asymmetric load: A ratio can be manually assigned to cause some backend servers to get a greater share of the workload than others. This is sometimes used as a crude way to account for some servers being faster than others.
- Priority activation: When the number of available servers drops below a certain number, or load gets too high, standby servers can be brought online.
- SSL Offload and Acceleration: SSL applications can be a heavy burden on the resources of a Web Server, especially on the CPU and the end users may see a slow response (or at the very least the servers are spending a lot of cycles doing things they weren't designed to do). To resolve these kinds of issues, a Load Balancer capable of handling SSL Offloading in specialized hardware may be used. When Load Balancers are taking the SSL connections, the burden on the Web Servers is reduced and performance will not degrade for the end users.
- Distributed Denial of Service (DDoS) attack protection: load balancers can provide features such as SYN cookies and delayed-binding (the back-end servers don't see the client until it finishes its TCP handshake) to mitigate SYN flood attacks and generally offload work from the servers to a more efficient platform.
- HTTP compression: reduces amount of data to be transferred for HTTP objects by utilizing gzip compression available in all modern web browsers.
- TCP offload: different vendors use different terms for this, but the idea is that normally each HTTP request from each client is a different TCP connection. This feature utilizes HTTP/1.1 to consolidate multiple HTTP requests from multiple clients into a single TCP

socket to the back-end servers.

- TCP buffering: the load balancer can buffer responses from the server and spoon-feed the data out to slow clients, allowing the server to move on to other tasks.
- Direct Server Return: an option for asymmetrical load distribution, where request and reply have different network paths.
- Health checking: the balancer will poll servers for application layer health and remove failed servers from the pool.
- HTTP caching: the load balancer can store static content so that some requests can be handled without contacting the web servers.
- Content Filtering: some load balancers can arbitrarily modify traffic on the way through.
- HTTP security: some load balancers can hide HTTP error pages, remove server identification headers from HTTP responses, and encrypt cookies so end users can't manipulate them.
- Priority queuing: also known as rate shaping, the ability to give different priority to different traffic.
- Content aware switching: most load balancers can send requests to different servers based on the URL being requested.
- Client authentication: authenticate users against a variety of authentication sources before allowing them access to a website.
- Programmatic traffic manipulation: at least one load balancer allows the use of a scripting language to allow custom load balancing methods, arbitrary traffic manipulations, and more.
- Firewall: direct connections to backend servers are prevented, for network security reasons.
- Intrusion Prevention System: offer application layer security in addition to network/-transport layer offered by firewall security.

2.4 Load Balancing In Telecommunications

Load balancing can be useful when dealing with redundant communications links. For example, a company may have multiple Internet connections ensuring network access even if one of the connections should fail.

A failover arrangement would mean that one link is designated for normal use, while the second link is used only if the first one fails. With load balancing, both links can be in use all the time. A device or program decides which of the available links to send packets along, being careful not to send packets along any link if it has failed. The ability to use multiple links simultaneously increases the available bandwidth.

Major telecommunications companies have multiple routes through their networks or to external networks. They use more sophisticated load balancing to shift traffic from one path to another to avoid network congestion on any particular link, and sometimes to minimize the cost of transit across external networks or improve network reliability.

2.5 Relationship With Failover

Load balancing is often used to implement failover the continuation of a service after the failure of one or more of its components. The components are monitored continually (e.g., web servers may be monitored by fetching known pages), and when one becomes non-responsive, the load balancer is informed and no longer sends traffic to it. And when a component comes back on line, the load balancer begins to route traffic to it again. For this to work, there must be at least one component in excess of the service's capacity. This is much less expensive and more flexible than failover approaches where a single "live" component is paired with a single "backup" component that takes over in the event of a failure. Some types of RAID systems can also utilize hot spare for a similar effect.

2.6 Persistence

An important issue when operating a load-balanced service is how to handle information that must be kept across the multiple requests in a user's session. If this information is stored locally on one backend server, then subsequent requests going to different backend servers

would not be able to find it. This might be cached information that can be recomputed, in which case load-balancing a request to a different backend server just introduces a performance issue...

One solution to the session data issue is to send all requests in a user session consistently to the same backend server. This is known as ”persistence” or ”stickiness”. A significant downside to this technique is its lack of automatic failover: if a backend server goes down, its per-session information becomes inaccessible, and any sessions depending on it are lost. The same problem is usually relevant to central database servers.

Assignment to a particular server might be based on a user name, client IP address, or random assignment. Owing to DHCP, Network Address Translation, and web proxies, the client's IP address may change across requests, and so this method can be somewhat unreliable. Random assignments must be remembered by the load balancer, which creates a storage burden. If the load balancer is replaced or fails, this information can be lost, and assignments may need to be deleted after a time-out period or during periods of high load to avoid exceeding the space available for the assignment table. The random assignment method also requires that clients maintain some state, which can be a problem, for example when a web browser has disabled storage of cookies. Sophisticated load balancers use multiple persistence techniques to avoid some of the shortcomings of any one method.

Another solution is to keep the per-session data in a database. Generally this is bad for performance since it increases the load on the database: the database is best used to store information less transient than per-session data. To prevent a database from becoming a single point of failure, and to improve scalability, the database is often replicated across multiple machines, and load balancing is used to spread the query load across those replicas. Microsoft's ASP.net State Server technology is an example of a session database. All servers in a web farm store their session data on State Server and any server in the farm can retrieve the data.

Fortunately there are more efficient approaches. In the very common case where the client is a web browser, per-session data can be stored in the browser itself. One technique is to use a browser cookie, suitably time-stamped and encrypted. Another is URL rewriting. Storing session data on the client is generally the preferred solution: then the load balancer is free to pick any back end server to handle a request. However, this method of state-data handling is not really suitable for some complex business logic scenarios, where session state payload is very big or recomputing it with every request on a server is not feasible, and URL rewriting has major security issues, since the end-user can easily alter the submitted URL and thus change session streams.

2.7 WAPT

Web Application Load Stress Performance Testing or WAPT 7.1 is a load and stress testing tool that provides you with an easy-to-use, consistent and cost-effective way of testing web sites, web servers, and intranet applications with web interfaces. You may test and analyze the performance characteristics under various load conditions to find bottlenecks of your web applications. WAPT has a set of features to test web sites with dynamic content and secure HTTPS pages. It provides informative test results through descriptive graphs and reports.

Today thousands of businesses worldwide face the challenge of establishing their web presence - a goal difficult to achieve without efficient web site development and testing tools. Why load and stress testing is so important? Most performance issues arise only when the server is stressed with a high user load. This means that you should perform load testing to know how many concurrent visitors your site can serve flawlessly. It can be difficult to organize such testing without the help of a group of real users. The right way is to use advanced automatic load and stress testing tools. WAPT can simulate up to several thousands real users to check the performance of your site and find any bottlenecks.

WAPT is designed for Microsoft Windows 2000/XP/2003/Vista/2008/Win7. It is competitively priced and does not require expensive hardware to run.

CHAPTER 3
Proposed System

3.1 Development and plan

Our project uses evolutionary process model. The Evolutionary Models take the concept of evolution into the engineering paradigm.[5] Therefore Evolutionary Models are iterative. They are built in a manner that enables software engineers to develop increasingly more complex versions of the software. Web load balancer is such a software that is iterative and can be developed to more complex versions.

3.1.1 The Incremental Model

The Incremental Model combines elements of the Linear Sequential Model (applied repetitively) with the iterative philosophy of prototyping. When an Incremental Model is used, the first increment is often the core product. The subsequent iterations are the supporting functionalities or the add-on features that a customer would like to see. More specifically, the model is designed, implemented and tested as a series of incremental builds until the product is finished.

Advantages

1. It is useful when staffing is unavailable for the complete implementation.
2. Can be implemented with fewer staff people.
3. If the core product is well received then the additional staff can be added.
4. Customers can be involved at an early stage.
5. Each iteration delivers a functionally operational product and thus customers can get to see the working version of the product at each stage.

3.1.2 The Spiral Model

The Spiral Model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the Linear Sequential Model. Using the Spiral Model the software is developed in a series of incremental releases. Unlike the Iteration Model where in the first product is a core product, in the Spiral Model the early iterations could result in a paper model or a prototype. However, during later iterations more complex functionalities could be added.

A Spiral Model, combines the iterative nature of prototyping with the controlled and systematic aspects of the Waterfall Model, therein providing the potential for rapid development of incremental versions of the software. A Spiral Model is divided into a number of framework activities, also called task regions. These task regions could vary from 3-6 in number and they are: Customer Communication - tasks required to establish effective communication between the developer and customer. Planning - tasks required to define resources, timelines and other project related information /items. Risk Analysis - tasks required to assess the technical and management risks. Engineering - tasks required to build one or more representation of the application. Construction and Release - tasks required to construct, test and support (eg. Documentation and training). Customer evaluation - tasks required to obtain periodic customer feedback so that there are no last minute surprises.

Advantages of spiral model

1. Realistic approach to the development because the software evolves as the process progresses.
2. The developer and the client better understand and react to risks at each evolutionary level.
3. The model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage of the evolutionary development.
4. It maintains a systematic stepwise approach, like the classic waterfall model, and also incorporates into it an iterative framework that more reflect the real world.

Disadvantages of spiral model

1. One should possess considerable risk-assessment expertise
2. It has not been employed as much proven models (e.g. the Waterfall Model) and hence may prove difficult to sell to the client.

CHAPTER 4

System Requirements Specification

4.1 Software Requirements

1. Windows 7/NT/XP
2. Java Runtime Environment v1.6.0_18
3. JEE SDK v6
4. Java IDE such as Eclipse/Netbeans
5. SQL

4.2 Hardware Requirements

1. Dual core processor
2. Hard disk 256Gb
3. Ethernet Port with Gigabit lan
4. RAM 1Gb
5. Standard monitor
6. Mouse
7. Keyboard

CHAPTER 5
Design & Analysis

5.1 System Analysis

This section presents an analysis of the project in terms of the module elaboration and effort distribution.

5.1.1 Module breakup

This section presents the module break-up.

5.1: Module Description

Module	Description
Front design	develops an interface for the client
Web load balancer implementation	coding web load balancer
Log creation	creates database containing client request details
Server implementation n application distribution	server accepts client request;sends requested data

5.1.2 Member effort

This section presents each member's effort in the team. The work-hours are also mention here alongside the module assigned.

You could use a table like the following to represent the work effort allocation.

5.2: Module Allocation

#	Task	Estimated Effort	Start Date	End Date	Person
1	Front end design	(2hrs)	(1/12/2010)	(20/12/2010)	Shaheen
2	W.L.B implementation	(2hrs)	(20/2/2011)	(28/3/2011)	Firose
3	Log creation	(2hrs)	(1/3/2011)	(28/3/2011)	Manya
4	Server implementation	(2hrs)	(30/3/2011)	(25/4/2011)	Vysakh
5	Application distribution	(2hrs)	(15/4/2011)	(25/4/2011)	Shaheen

5.2 System Design

5.2.1 Use Case Models / Flow Diagrams

The usecase model of web load balancer in shown below.

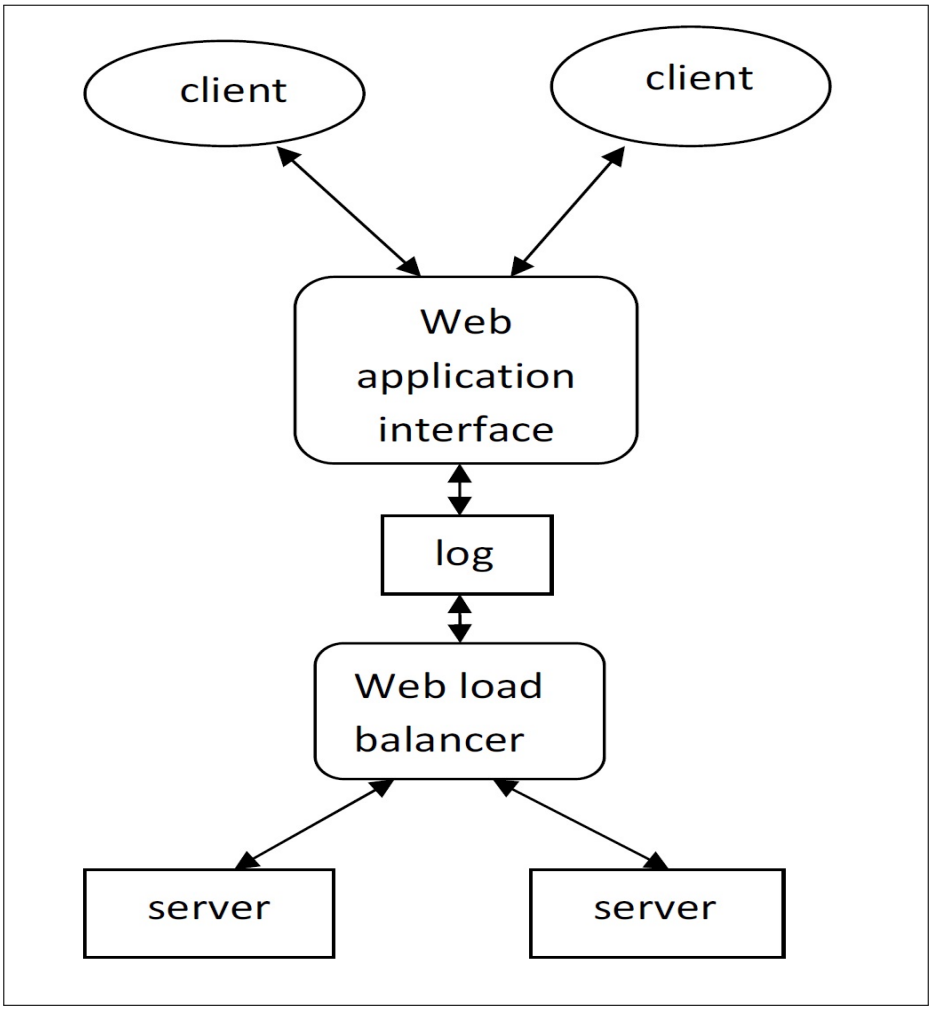

Fig. 5.1: Usecase model for web load balancer

CHAPTER 6
Implementation

6.1 Introduction

6.1.1 User Module

In the user side, the user gives the search data in the search box.On request, the search result will be displayed in the result box.

6.1.2 Web load balancer Module

In the web load balancer module, total processes running in each connected server is displayed. The least loaded server which is used for request data retrieval is also shown in the controller or web load balancer. Log details option can also be chosen from the load balancer.

6.1.3 Log details Module

Details of which data was requested and when it was requested are stored in the log file.

6.1.4 Server Module

Details of all the processes running in the server will be displayed.

6.1.5 Application Module

This module consists of a database of dictionary details.These details are distributed in all the connected servers.When the user requests for the word details,the requested data will be displayed in the client computer.

6.2 Screenshots

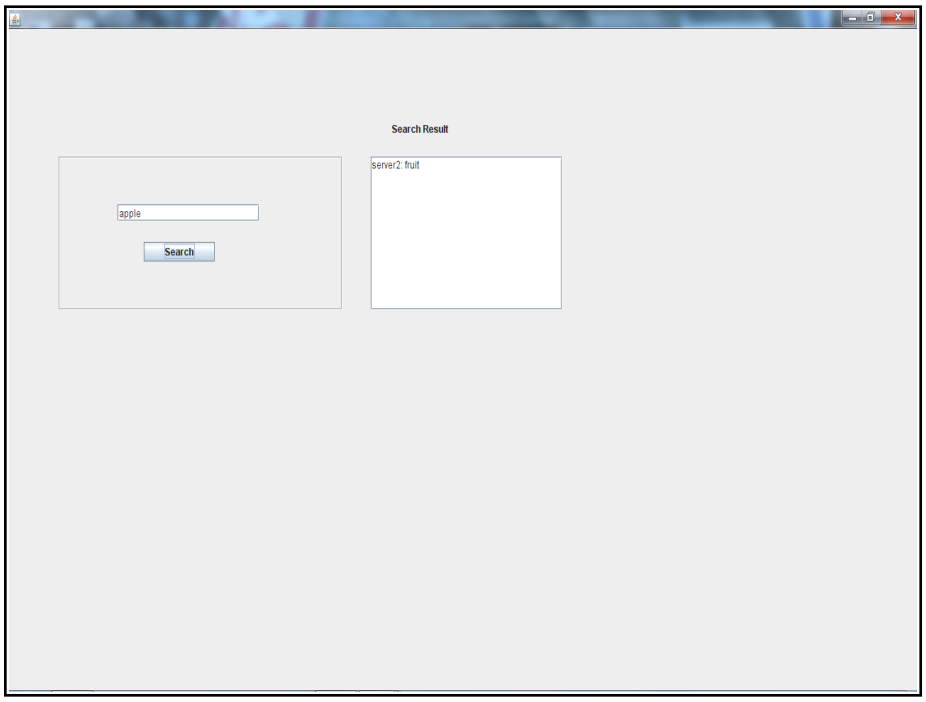

Fig. 6.1: Screenshot 1-user

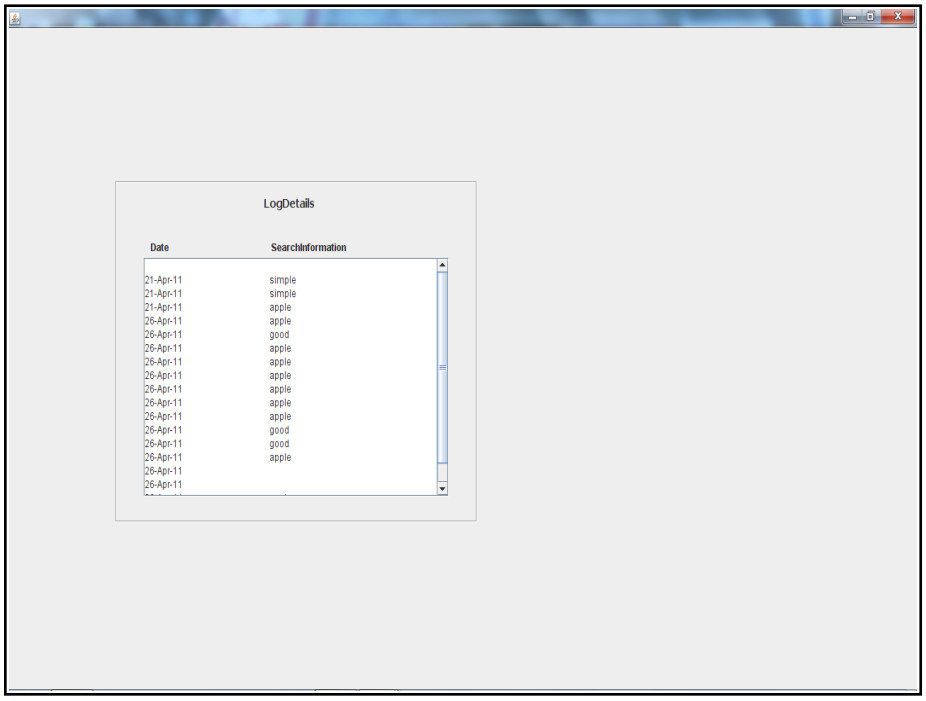

Fig. 6.2: Screenshot 2-log details

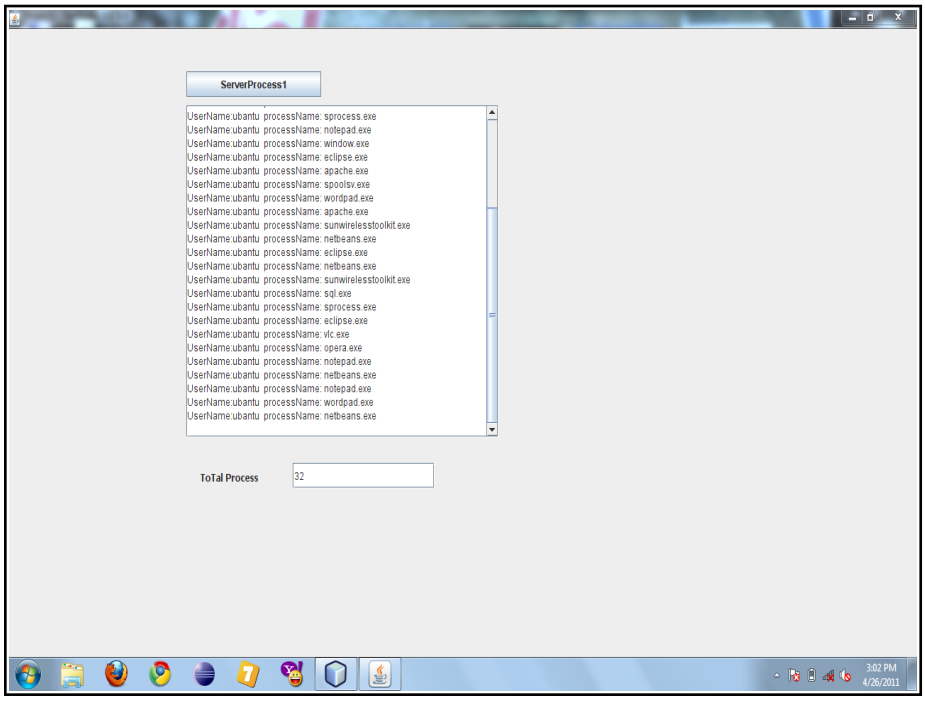

Fig. 6.3: Screenshot 3-server process

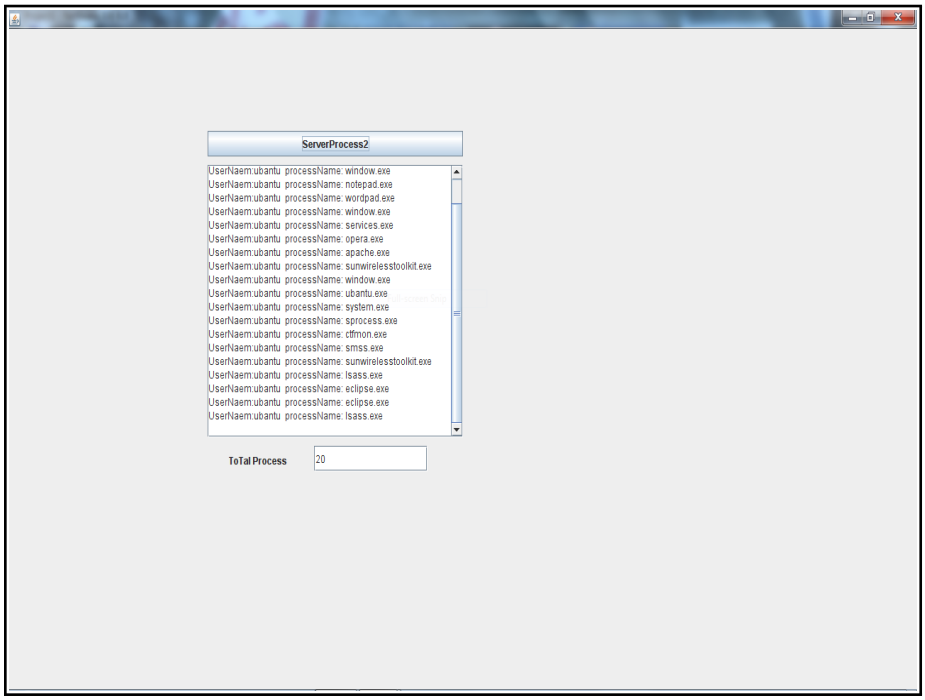

Fig. 6.4: Screenshot 4-controller

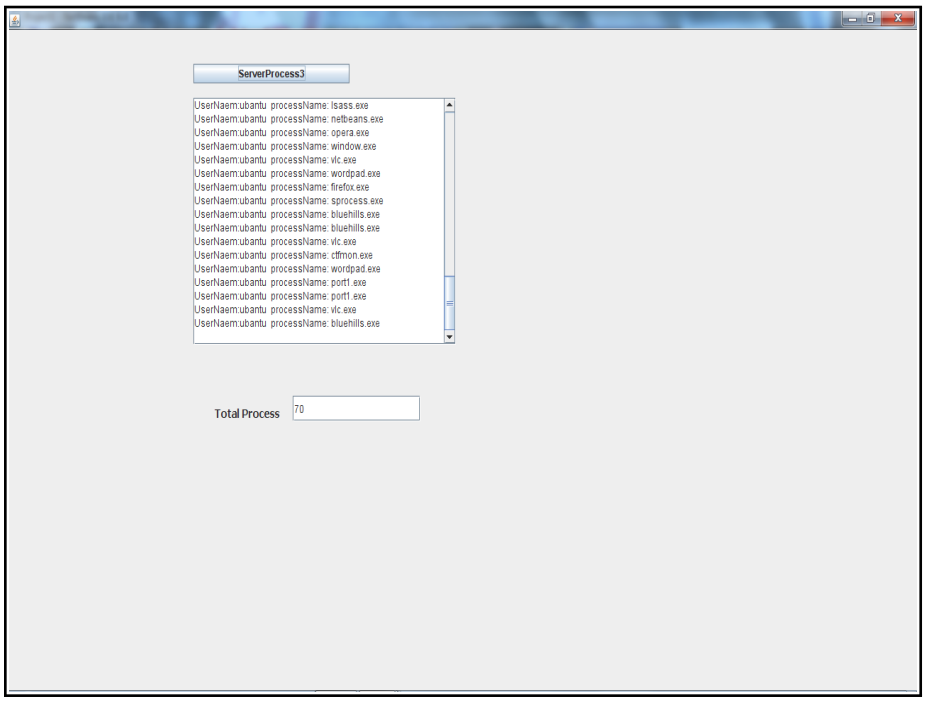

Fig. 6.5: Screenshot 3-server process

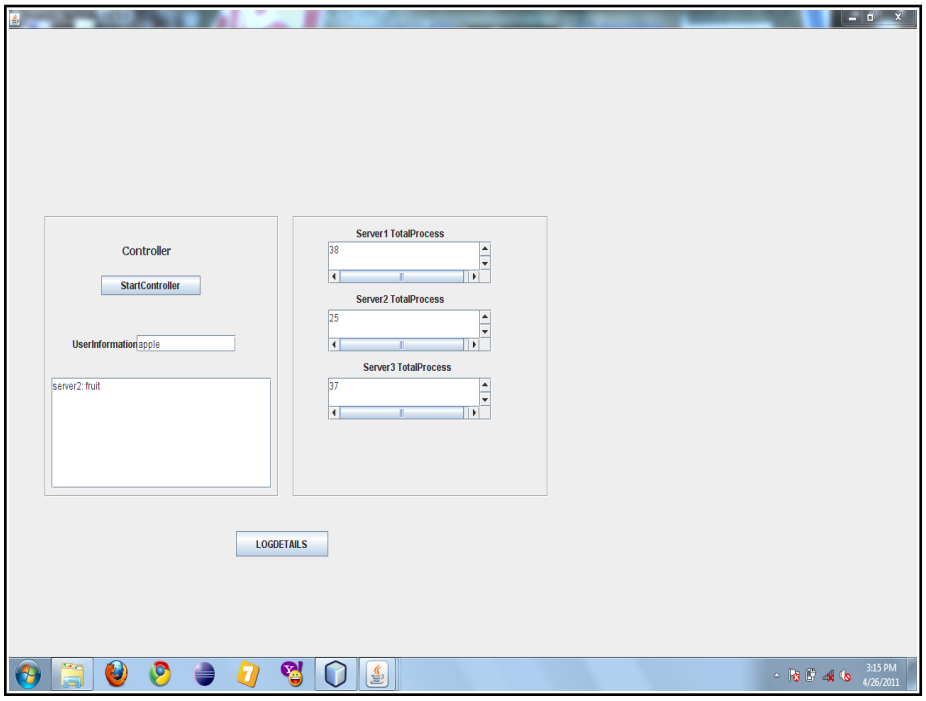

Fig. 6.6: Screenshot 4-controller

6.3 Pseudo codes

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;


public class User extends javax.swing.JFrame {

    /** Creates new form User */
    public User() {
        initComponents();
        jTextArea1.setText("search information:");
    }


    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
    desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();

        setDefaultCloseOperation
        (javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setMinimumSize(new java.awt.Dimension(950, 750));
        getContentPane().setLayout(null);

        jPanel1.setBorder
        (javax.swing.BorderFactory.createEtchedBorder());
```

```
jPanel1.setLayout(null);
jPanel1.add(jTextField1);
jTextField1.setBounds(83, 56, 200, 20);

jButton1.setText("Search");
jButton1.addMouseListener
(new java.awt.event.MouseAdapter() {
 public void mouseClicked
 (java.awt.event.MouseEvent evt) {
        jButton1MouseClicked(evt);
    }
    public void mouseExited
    (java.awt.event.MouseEvent evt) {
        jButton1MouseExited(evt);
    }
});
jButton1.addActionListener
(new java.awt.event.ActionListener() {
    public void actionPerformed(
    java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
jPanel1.add(jButton1);
jButton1.setBounds(120, 100, 100, 23);

getContentPane().add(jPanel1);
jPanel1.setBounds(70, 150, 400, 180);

jLabel1.setText("Search Result");
getContentPane().add(jLabel1);
jLabel1.setBounds(540, 110, 120, 14);

jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);
```

```
        getContentPane().add(jScrollPane1);
        jScrollPane1.setBounds(510, 150, 270, 180);

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed
    (java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        run();
        /* String str1=jTextField1.getText();
        try
        {
        ServerSocket ss=new ServerSocket(9000);
        Socket cs=ss.accept();
        DataInputStream din=new
        DataInputStream(cs.getInputStream());
        DataOutputStream dout=new
        DataOutputStream(cs.getOutputStream());
        dout.writeUTF(str1);
        String str2=din.readUTF();
        jTextArea1.setText(str2);
        }
        catch(Exception e)
        {
        e.printStackTrace();
        }*/
    }//GEN-LAST:event_jButton1ActionPerformed

    private void jButton1MouseExited
    (java.awt.event.MouseEvent evt)
    {//GEN-FIRST:event_jButton1MouseExited
        // TODO add your handling code here:

    }//GEN-LAST:event_jButton1MouseExited
```

```
private void jButton1MouseClicked
(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_jButton1MouseClicked
    // TODO add your handling code here:
    server2();
}//GEN-LAST:event_jButton1MouseClicked
public void run() {
    String str1 = jTextField1.getText();
    try {
        ServerSocket ss = new ServerSocket(9000);
        Socket cs = ss.accept();
        DataInputStream din = new DataInputStream
        (cs.getInputStream());
        DataOutputStream dout = new DataOutputStream
        (cs.getOutputStream());
        dout.writeUTF(str1);
        cs.close();
        ss.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {
            new User().setVisible(true);
        }
    });
}
public void server2()
{
     try {
```

```
            ServerSocket ss = new ServerSocket(7000);
            System.out.println("server runnining port 7000");
            Socket cs = ss.accept();
            DataInputStream din = new DataInputStream
            (cs.getInputStream());
            DataOutputStream dout = new DataOutputStream
            (cs.getOutputStream());
            String str2 = din.readUTF();
            jTextArea1.setText(str2);
            cs.close();
            ss.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    // Variables declaration -
    do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration//GEN-END:variables
}
//controller
private void jButton1MouseExited
(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_jButton1MouseExited
        // TODO add your handling code here:
        String str1=jTextArea1.getText();
        String str2=jTextArea2.getText();
        String str3=jTextArea3.getText();
        Long s1=Long.valueOf(str1);
        Long s2=Long.valueOf(str2);
        Long s3=Long.valueOf(str3);
        if(s1<s2&&s1<s3)
```

```
        {
       client1();
        }
        else if(s2<s1&&s2<s3)
        {
            client2();
        }
        else
        {
            client3();
        }
        client4();

    }//GEN-LAST:event_jButton1MouseExited

    private void jButton2ActionPerformed
    (java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_jButton2ActionPerformed
        // TODO add your handling code here:

                new logfile().setVisible(true);
                logfile ll=new logfile();
                ll.display();


    }//GEN-LAST:event_jButton2ActionPerformed
   public void client1()
   {

          try
          {
           Socket cs=new Socket("127.0.0.1",8001);
           DataInputStream din=new
           DataInputStream(cs.getInputStream());
           DataOutputStream dout=new
           DataOutputStream(cs.getOutputStream());
           dout.writeUTF(jTextField1.getText());
```

```
            String str1=din.readUTF();
            jTextArea4.append(str1);

                }
            catch(Exception e)
            {
                e.printStackTrace();
            }
    }
    public void client2()
    {

            try
            {
             Socket cs=new Socket("127.0.0.1",8002);
             DataInputStream din=new
             DataInputStream(cs.getInputStream());
             DataOutputStream dout=new
             DataOutputStream(cs.getOutputStream());
             dout.writeUTF(jTextField1.getText());
             String str1=din.readUTF();
             jTextArea4.append(str1);

     }
            catch(Exception e)
            {
                e.printStackTrace();
            }
    }
    public void client3()
    {

            try
            {
             Socket cs=new Socket("127.0.0.1",8003);
             DataInputStream din=new
             DataInputStream(cs.getInputStream());
```

```
            DataOutputStream dout=new
            DataOutputStream(cs.getOutputStream());
            dout.writeUTF(jTextField1.getText());
            String str1=din.readUTF();
            jTextArea4.append(str1);

     }
           catch(Exception e)
           {
               e.printStackTrace();
           }
    }
    public void client4()
    {

           try
           {
            Socket cs=new Socket("127.0.0.1",7000);
            DataInputStream din=new
            DataInputStream(cs.getInputStream());
            DataOutputStream dout=new
            DataOutputStream(cs.getOutputStream());
            dout.writeUTF(jTextArea4.getText());
     }
           catch(Exception e)
           {
               e.printStackTrace();
           }
    }

  public void add() throws Exception
        {

     DateFormat formatter ;
                 Date date ;
               formatter = new SimpleDateFormat("dd-MMM-yy");
               date = new Date();
```

```
            String s = formatter.format(date);
              String str1=jTextField1.getText();
              System.out.println("Hai");
              Class.forName("com.mysql.jdbc.Driver").
              newInstance();
      System.out.println("driver ");
      Connection conn = (Connection)
      DriverManager.getConnection("jdbc:mysql://
      localhost:3306/details","root","root");
      System.out.println("connected");
      Statement statement = conn.createStatement();
      statement.executeUpdate("INSERT INTO log1
      VALUES ('"+s+"','"+str1+"')");
      }
  public static void main(String args[]) {
              new Controller().setVisible(true);
  }
  // Variables declaration - do not modify
  //GEN-BEGIN:variables
  private javax.swing.JButton jButton1;
  private javax.swing.JButton jButton2;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel2;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JLabel jLabel4;
  private javax.swing.JLabel jLabel5;
  private javax.swing.JPanel jPanel1;
  private javax.swing.JPanel jPanel2;
  private javax.swing.JScrollPane jScrollPane1;
  private javax.swing.JScrollPane jScrollPane2;
  private javax.swing.JScrollPane jScrollPane3;
  private javax.swing.JScrollPane jScrollPane4;
  private javax.swing.JTextArea jTextArea1;
  private javax.swing.JTextArea jTextArea2;
  private javax.swing.JTextArea jTextArea3;
  private javax.swing.JTextArea jTextArea4;
  private javax.swing.JTextField jTextField1;
```

```
    // End of variables declaration//GEN-END:variables
class client1 extends Thread {
     public void run()
     {
        client1();
     }
     public void client1()
     {
         try
         {
          Socket cs=new Socket("127.0.0.1",9001);
          DataInputStream din=new
          DataInputStream(cs.getInputStream());
          DataOutputStream dout=new
          DataOutputStream(cs.getOutputStream());
          String str1=din.readUTF();
          jTextArea1.setText(str1);
    }
         catch(Exception e)
         {
             e.printStackTrace();
         }
       }
}
 class client2 extends Thread{
      public void run()
     {
        client2();
     }
     public void client2()
     {
         try
         {
          Socket cs=new Socket("127.0.0.1",9002);
          DataInputStream din=new
          DataInputStream(cs.getInputStream());
          DataOutputStream dout=new
```

```
            DataOutputStream(cs.getOutputStream());
            String str1=din.readUTF();
            jTextArea2.setText(str1);

          }
          catch(Exception e)
          {
              e.printStackTrace();
          }
       }
}
class client3 extends Thread
{
       public void run()
     {
        client3();
     }
     public void client3()
     {
         try
       {
            Socket cs=new Socket("127.0.0.1",9003);
            DataInputStream din=new
            DataInputStream(cs.getInputStream());
            DataOutputStream dout=new
            DataOutputStream(cs.getOutputStream());
            String str1=din.readUTF();
            jTextArea3.setText(str1);
       }
          catch(Exception e)
          {
              e.printStackTrace();
          }
       }
   }
}
```

```
//Log file
public class logfile extends javax.swing.JFrame {

    /** Creates new form logfile */
    public logfile() {
        initComponents();
        display();
    }
    public void display()
    {
         try
        {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        System.out.println("Connection Registered");
        Connection con = DriverManager.getConnection
        ("jdbc:mysql://localhost:3306/details","root","root");
        Statement st = (Statement) con.createStatement();
        //ResultSet rs=st.executeQuery(" select date1,count(*)
         from log1 group by date1");
        ResultSet rs=st.executeQuery(" select *from log1");

        while(rs.next())
        {                String s4=rs.getString(1);
                String s2=rs.getString(2);
           jTextArea1.append("\n"+""+s4+"\t\t"+s2);
        }
        }
        catch(Exception e)
        {
                e.printStackTrace();
        }
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
    desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {
```

```
jPanel1 = new javax.swing.JPanel();
jLabel3 = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();

setDefaultCloseOperation(javax.swing.
WindowConstants.EXIT_ON_CLOSE);
setMinimumSize(new java.awt.Dimension(950, 750));
getContentPane().setLayout(null);

jPanel1.setBorder(javax.swing.
BorderFactory.createEtchedBorder());
jPanel1.setLayout(null);

jLabel3.setFont(new java.awt.Font
("Tahoma", 1, 14)); // NOI18N
jLabel3.setText("LogDetails");
jPanel1.add(jLabel3);
jLabel3.setBounds(210, 10, 100, 30);

jLabel1.setText("Date");
jPanel1.add(jLabel1);
jLabel1.setBounds(50, 70, 50, 14);

jLabel2.setText("SearchInformation");
jPanel1.add(jLabel2);
jLabel2.setBounds(220, 70, 110, 14);

jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);

jPanel1.add(jScrollPane1);
jScrollPane1.setBounds(40, 90, 430, 280);
```

```
        getContentPane().add(jPanel1);
        jPanel1.setBounds(150, 180, 510, 400);

        pack();
    }// </editor-fold>//GEN-END:initComponents

    /**
    * @param args the command line arguments
    */


    // Variables declaration -
    do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration
    //GEN-END:variables

}

// server
private void jButton1ActionPerformed
(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        Random r1 = new Random();
        int x = r1.nextInt(100);
        String x1 = Integer.toString(x++);
        System.out.println(x1);
        try
        {
            ServerSocket ss=new ServerSocket(9001);
            System.out.println("server
```

```
            waiting for client connection...");
            Socket cs=ss.accept();
            DataInputStream din=new
            DataInputStream(cs.getInputStream());
           DataOutputStream dout=new
           DataOutputStream(cs.getOutputStream());
           dout.writeUTF(jTextField1.getText());
          cs.close();
           ss.close();
           server1();

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }//GEN-LAST:event_jButton1ActionPerformed
public void server1()
{
     try
        {
            ServerSocket ss=new ServerSocket(8001);
            System.out.println("server waiting
             for client connection 8001...");
            Socket cs=ss.accept();
            DataInputStream din=new
            DataInputStream(cs.getInputStream());
            DataOutputStream dout=new
            DataOutputStream(cs.getOutputStream());
            String str1=din.readUTF();
            System.out.println(str1);
                System.out.println("Hai");
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                Connection con = (Connection)
                 DriverManager.getConnection("jdbc:mysql://
                 localhost:3306/details", "root", "root");
                Statement st=con.createStatement();
```

```
                ResultSet rs=st.executeQuery("select
                 * from word where text='"+str1+"'");
                String f2=null;
                while(rs.next())
                {
                        f2=rs.getString(2);
                }
            dout.writeUTF(f2);

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
}
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ServerProcess1().setVisible(true);
            }
        });
    }
```

CHAPTER 7
Testing & Maintenance

7.1 Tests

In general,it is impossible to test it under all possible operating conditions.Thus,it is necessary to find suitable test cases that provide evidence to give us confidence that the desired behaviour will be exhibited even in cases that have not been tested. This is often a difficult and sometimes even impossible,job. Future more,in the case of software testing, the usual analogies between traditional engineering fields and software engineering fail to provide useful suggestions. Software testing is an important section in software development. It is critical element of software assurance. Good testing is that which has high probability of finding the error which is undiscovered. System testing is based on the logical reasoning that if all parts of the system are correct, the goal will be successfully achieved. Inadequate testing or no testing leads to errors, that may appear after the implementation of the system. In the testing procedure the following procedures are to be followed.

7.1.1 Unit Testing

This involves test carried out in modules or program which makes up a system. This is also called program testing. The program should be tested for correctness of logic applied and should detect errors in the code. Web load balancer was coded so as to find out which server has less load. It compares the loads of the servers and and sends the user request to the least loaded server. The log module stores the details of the user requests. The GUI phase creates a user friendly interface for the client.

7.1: Unit test chart

No	Unit Name	Test Status
1	GUI module	Complete
2	Web load balancer	Complete
3	Log creation	Complete
4	Application implementation	Complete

7.1.2 Integration Testing

When the unit tests are satisfactorily conducted the system as a complete entity must be tested.In our project,when the user requests for a data through the interface,the load balancer executes and checks which one of the three servers has the least load and directs the request to the least loaded server. The server retrieves the data to the user.

7.1.3 Validation Testing

During this test we got the final assurance that the software meets all the functional, behavioural and performance requirements. Validation succeeded when the software functions in a manner in which user wishes. Validation refers to the process of using software in live environment in order to find errors. During the course of validation the system failure may occur and sometimes the coding has to be changed according to the requirement. Thus the feedback from the validation phase had generally produced changes in the software.

7.1.4 User Acceptance Testing

Acceptance testing refers to the acceptance of data into the system for processing. The acceptance test contributes to the consistency and smooth working of the system. The system under consideration is tested for the user at a time of developing and making changes whenever required.

7.2 Maintenance

The term maintenance is commonly used to refer to the modifications that are made to a software system after its initial phase. Maintenance involves changing the software to improve some of its qualities. Changes are due to the need to modify the functions,improve the performance of the application,making it easier to use etc. An application that consists of well-designed modules is easier to analyse and repair than a monolithic one. We have to choose the right interfaces that avoid complex interconnections and interactions among modules. The application is repairable; its defects can be corrected with a reasonable amount of work. Maintenance involves adjusting the application to changes in the environment. The

application changes as the environment in which it is embedded is changed. Repairability is achieved by using the standard parts that can be easily replaced. Since the system is modular, it is easy to analyse and repair. Many modules in the system are reusable.

CHAPTER 8
Conclusion

8.1 Introduction

Load balancing is an already implemented concept. In computer networking, load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Load balancing is extremely important and it is fundamental to the operational success of some of the most recognized, high-traffic Websites visited today. [2]

Our project uses evolutionary process model. The Evolutionary Models take the concept of evolution into the engineering paradigm. Therefore Evolutionary Models are iterative. They are built in a manner that enables software engineers to develop increasingly more complex versions of the software. Web load balancer is such a software that is iterative and can be developed to more complex versions.

In our project,when the user requests for a data through the interface,the load balancer executes and checks which one of the three servers has the least load by comparing the loads of the servers and directs the request to the least loaded server. The server retrieves the data to the user. At the server side, a dictionary as a database is set as the application. So when the user requests a word, the meaning of that requested word is retrieved to the client.

8.2 Future work

Our project can be modified at the application phase. Currently in our project, a database of dictionary is set as application. As future development, the web page retrieval can be implemented. So each server will have the website and its complete details implemented in it. The user when requests for the web page, the least loaded server retrieves the web page to the client.

Modification can be done at the client phase also.In our project, a single client makes random requests to the server. As future scope, multiple clients can be implemented.

REFERENCES

[1] xxx. (2001, Jun) titkle. [Online]. Available: www.mysite.com

[2] J. Fry and M. Langhammer, "Fpgas lower costs for rsa cryptography." [Online]. Available: http://www.design-reuse.com/articles/6358/fpgas-lower-costs-for-rsa-cryptography.html

[3] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, 2nd ed. Addison-Wesley, 1981, vol. 2.