UNIVERSITY OF OSLO
Department of Informatics

Performance Measurement of Web Services Linux Virtual Server

Muhammad Ashfaq
Oslo University College

May 19, 2009

Performance Measurement of Web Services Linux Virtual Server

Muhammad Ashfaq
Oslo University College

May 19, 2009

Abstract

With the rapid expansion in the use of internet services such as web browsing, mails, audio and video files downloading, servers' needs to manage with greater strain resources and actions. Demand for total number of clients supported by the servers has increased significantly. With a continues increase in total number of users and as a result escalating work load makes businesses uncertain about their actions with the passage of time. In addition rapid response and 24 hours availability becoming mandatory necessities for many big critical businesses applications as a result, the necessity of providing support for extremely expandable, sustainable and available services is becoming crucial.

Linux Virtual Server is the best solution for providing businesses such critical services. Linux Virtual Server is an open source tool, used to establish expandable, sustainable and highly available server using a number of real servers and a front end Director. Linux Virtual Server has the ability to balance Load of a number of network services amongst several real servers using different scheduling algorithms and packet forwarding methods which suits bests for services and hardware limitations.

In this project a web services Linux Virtual Server will be established using NAT (Network Address Translation) packet forwarding method and using three different scheduling algorithms, Round Robin, Weighted Round Robin and least Connection. With each scheduling algorithm number of test setup will be conducted by varying number of real servers used in the Linux Virtual Server Cluster, varying HTTP traffic through using different size of data files downloaded from the real servers and also varying number of request send per second.

Acknowledgements

I am highly thankful for my supervisor Professor Tore M. Jonassen providing guidelines throughout in writing my final thesis. I am especially thankful to him for providing assistance in lab experiments and data collections. I am very thankful for Professor Mark Burgess, Oslo University College and University of Oslo for giving me opportunity for studying Master degree in Networks and System Administration. I am also thankful to my parents and all of my friends who supported and encouraged me for studing Master of Networks and System Adminstration.

Contents

Chapter 1

Introduction

1.1 Introduction to Linux Virtual Server

With the enlargement of the internet, servers are required to manage with larger burden of resources and actions. The possible number of clients that a server has to support has significantly enlarged. With the growing figure of users and the escalating amount of work load, business a lot uncertain about how their proceedings raise over time. Additionally, fast reply and twenty-four hour availabilities are obligatory necessities for the serious undertaking big business applications, as web sites struggle for providing users the most excellent way of access. As a result, the necessity of providing support for extremely expandable, sustainable and available services is crucial.

Linux Virtual Server is an extension to the core Linux operating system that allows a number of servers to provide network services as a single virtual server[2]. The combined skills of the entity hosts, or real servers, constitute the LVS clusters are often beyond those of any of the components as an individual unit. Power is also increased because the services are supported by several computers, and thus avoid the only point of breakdown one can face with a single server.

1.2 Linux Virtual server Overview

Linux virtual server is a very expandable, variable and reliable server comes together with a set of real servers. It has the ability to equally distribute load on real servers through a Load balancer working on the Linux operating system.

Construction of server cluster is fully understandable and evident to end users, that when users connect to the cluster system, they assume as if they were connected to a single very fast and powerful server. Load balancer and actual real servers both can exist on same local area network or on different local area network over a WAN [1]. Load balancer uses different techniques to transmit client request among different real servers and make corresponding services on the cluster to come into view as a service on a single IP address. Sending requests to real servers can be accomplished by three IP Load balancing methods and TCP/IP stack of the Linux Kernel can be develop to support these techniques.

Linux virtual server makes us possible to clearly and evidently add or remove a node in the cluster which ensures scalability of the system. The system must also be

reliable and reliable system insures 100 percent availability of the services. This can greatly be achieved through Linux Virtual Server by discovering the existence of node or service failures and reconfiguring the system properly.

Linux Virtual Server is implemented as layer-4-switch in the Linux kernel [4]. This means that TCP and UDP manage to equally distribute work load between various physical servers. Thus it provides an approach to balance Web services ahead of a particular host. HTTP and HTTPS transfer the web services is most likely the frequent use.

Linux Virtual Server is an incredibly elevated implementation. It is capable to hold more than 100,000 instantaneous links. It makes possible to easily distribute work load of 100Mbit Ethernet connection with cheap good hardware. Even it not only makes possible to equally distribute load of 1Gbit connection but also capable for higher-end standard machines. The one server solution is typically not adequate to treat with this powerfully growing load. The process of upgrading becomes composite, and it is the only point of breakdown.

Clusters of servers, linked by a speedy set of connections are rising as a feasible structural design for constructing highly scalable and highly accessible services [3]. This type of freely attached design construction is expandable, more cost-efficient and dependable than a strongly joined structural system. There are many issues and challenges which must be handled professionally to make the cluster servers, work successfully for expandable services. The solution for highly available and scalable services is Linux Virtual Server. As Linux Virtual Server directs incoming client requests to several servers which are sharing work burden equally, so it can be used to construct extremely expandable and highly accessible services.

Redirecting of clients incoming connection requests to the real servers take place through different scheduling methods [3]. This makes possible for the cluster services to come into view as virtual service with single unique IP address. For the clients connected from outside work together with cluster as a unique server. When ever needed cluster can be interacted and this will not affect clients and clients dont need any update. This architectural design makes possible to easily add or remove a server from cluster achieving expandability and scalability. The design can be make highly accessible by monitoring nodes and maintaining the system without affecting clients request.

1.3 Basic Operations of LVS

Linux Virtual server [5] is a quick and expandable technique for load-balancing of essential components of any Network, could be Web, e-mail application or services. Single point of failure services now with Linux virtual server, service can easily be configured as scalable and available.

Lets consider a plain situation, a website that is spread through round-robin DNS to try to balance load among several servers. Unluckily, this is only an estimated equilibrium, and if a server fails, there is no way for making service available. Down server IP address still used by the clients. This is the case where Linux virtual server without any difficulty wins progress. Linux virtual server uses Linux in such a way that it encapsulates many real servers at a back of one virtual server. As Linux virtual

server solution is implemented in kernel and kernel already manages the network layer. That makes it possible to direct links professionally without any burden of a separate running process or program.

IPVS (Linux IP Virtual Server) is one of the most popular applications of Linux Virtual Server. IPVS can be established in three different modes. The easy and simple way to setup IPVS is NAT Routing. This method uses Network Address translation. When request reaches the director, it rewrites the header information in such a way that destination of the packet will be one of real server in the cluster. After one of the real server handling request it sends reply packets back to the load balancer/ director and upon arriving director rewrites the header information and sets the destination address of the packet back to the source address of the client.

In NAT Routing method, Linux Virtual Server Router or Director serves as central point of all incoming clients requests and outgoing real servers response. It receives incoming clients requests and directs these requests to suitable backend real server. Real server processes these requests and sends response back to the LVS director, as for real server the source of request is LVS director. Its director responsibility to keep record of all incoming and outgoing traffic, as it as to change addresses of real server and clients in the packets header information and director uses network address translation to accomplish this job.

This procedure of NAT Routing is called IP masquerading as the real IP addresses of the real servers are unknown from the incoming clients requests [6]. In NAT Routing topology, we can have real servers any system having any kind ok operating system. As in NAT Routing director must have to process all incoming and outgoing request, so its the main disadvantage of this topology as it can become blockage in heavy network load.

Second mode for setting IPVS is direct routing. In direct routing method when packets arrives at load balancer/ director, it rewrites header information and changes destination hardware address to one of the real server hardware address in the cluster, as a result packets are thrown back to the director to be received by one of the real server node of the cluster.

In the distinctive direct routing Linux Virtual Server method, all request coming from the clients received by the LVS director or router on a virtual IP (VIP) [6]. The director or router upon arriving directs these requests to one of the real server. To accomplish this task director uses different scheduling algorithms. In direct routing real server is responsible for sending reply back directly to the clients. Direct routing methods achieves scalability in the sense that it reduces burden on the director of outgoing traffic and sends outgoing traffic direct to the appropriate client. But in NAT as all outgoing traffic passes through the director that can become a blockage during intense network load.

Constructing a LVS setup which uses direct routing provides better performance benefits compared to other LVS networking methodologies [6]. Direct routing makes possible for the real servers to analyze incoming packets and after processing request making intelligent decision to route reply packets straight to actual requesting client without consulting Linux Virtual Server director. This special characteristic of Direct Routing plays great role in performance by reducing the Linux Virtual Server work load.

The third mode is IP tunnelling, which encapsulates the original packet with a new

IP header that directs the packet to the node selected.

If a user has privilege to access a virtual service offered by the Linux Virtual server cluster then when client sends request for that service, a packet sent with a destination address of the virtual server arrives. The director which performs load balancing analyses the packet header to find port and end node address. If the end nodes address matches with the address of the virtual server then virtual server selects a real server from the cluster through a link scheduling algorithm. Load balancer adds every link into a hash table, which is responsible for maintaining all links record [1].

Then load balancer enclose packet into an IP header and transfers it to the selected server. Now load balancer has this connection stored in hash table, whenever packets come on this link and the selected server is stored in the hash table, it will enclose packet once again into header and transfers it to the appropriate server. Upon receiving enclosed packets server disclose packets and performs requesting task, and after handing out it transfers final results to the client on the basis of its entry in routing table. The hash table refreshes itself after the time out of each established link and removes entry from the hash table [3].

IP tunnels provide an exciting and cheap substitute to a WAN (Wide Area network). WAN's are often established, if it is necessary to make the network between different physical areas. Generally WAN is very costly to establish and operate. Its necessary to have high speed data links for establishing a Wide Area Network.

Even though the Internet today provides contact to almost everywhere in the world through virtual technology, but it has many problems related to authentication and protection. IP tunnel may be a solution to these problems.

An additional problem occurs when a site wants to share private data, for example confidential web server, backup of networks, or a different website. Typically, firewalls are used at every server location to hide the website behind that firewall so that external attempt for accessing private data can be stopped. In this way firewalls protect private data not to be shared.

In IP tunnelling encoded tunnels are transferred over the IP, which provides way to establish connection between two local area networks (LANS), this topology allows networks to share data having a firewall in between.

Any one of these solutions comes with a work distributed or balancer which makes services come into view to start on the load balancer, but basically these services are generated from one of the real server in the pool.

1.4 High Availability Services

Companies now a days needs nonstop availability, accessibility of data and applications within variety of operating systems, different kind of applications and within different hardware solutions as well as several data locations.

Companies depend increasingly on IP networks for central big business exercise. Highly available and accessible network is becoming more and more essential, because the system failure typically provides major efficiency, production and income losses for many companies. Increasing the system running time up, insists the use of most excellent and best practices and having a backup network design in combination with highly available technology in the network essentials. Businesses need to implement

the essential elements of highly available and highly accessible solution and implement design in such a way that availability and accessibility can be achieved by the company and serviceable Sites that focus on business accessibility [7].

Businesses network must be made highly available and highly accessible, to guarantee that their crucial and important running programs will always remain accessible and available. Improved accessibility and availability leads to increased production and possibly increase sales and greatly reduces production cost. The reliable system means that the system at every cost must perform certain tasks properly, whereas by available system means that the system must be running every time to response urgently. To meet 100 percentage availability and accessibility networks must be accessible and available at every minute, every second of every day and every month of the year.

High accessibility and high availability fulfils allot of businesses needs some are mentioned bellow. First of all accessibility and availability make sure that critical businesses applications will always be available. Main function of the companys Network is to assist the network services. If these services are not available, the company no longer functions appropriately. The highly available network makes sure that the companys mission-critical services are running, available and accessible.

Secondly availability plays great rule for the improvement of workers and clients pleasure and faithfulness. Downtime of any network application or service can lead to dissatisfaction between both workers and clients who tried to access those services. So providing network applications and services highly accessible and available contributes to the improvement and maintenance of pleasure and faithfulness of workers as well as customers.

Thirdly availability decreases immediate consultancy support cost of information technology which greatly improves efficiency and increases organizational success. Planning a network that ensures high accessibility and high availability of IT services, decreases businesses maintenance time to minimum level. It gives time to businesses for increasing productive services to increase profit. Fourthly availability plays great rule to minimize the economic loss. If network is not up and services are not available that can transform straight into profits loss for the company. Fifthly availability reduces lost production. If the services or applications are down, workers cant carry out their tasks efficiently. Production lost means higher costs for the company.

1.5 Overview of Linux Virtual Server scheduling

Linux Virtual Server ability to execute load distribution on the real server farm is one of the important feature. LVS is very flexible in load distributing. LVS administrators have flexibility to choose from a large number of scheduling algorithms solutions before constructing a LVS cluster [8].

Load balancing of LVS is better than minimum flexibility, like the hierarchical structure of DNS in Round-Robin DNS method and client system caching can direct towards bad work distribution. Even low-level filter of LVS router has great reward over the call forwarding at system level, since distributing work on the network packet filter level puts less processing cost and provides better scalability and accessibility. LVS scheduling used by the router or directive to remember the real servers and their

status and assigns a waited feature to an application or service request. The assignment of weights gives random priority to each system. With this type of scheduling makes achievable to construct a collection of real servers through a range of software as well as hardware solutions and makes possible for the director or router to equally distribute work on every individual server in the cluster.

The scheduling method for the LVS is provided by IPVS or IP virtual server. IPVS module actives switching of transport layer that is considered to work best with several servers using a unique address. IP Virtual Server constructs a table within kernel to effectively direct packets towards real servers. LVS router or director used this table to transfer requests from virtual address to one of real server. A routine named ipvsadm regularly adds and removes real servers of the cluster depending on which server is running and services are available.

1.6 Scheduling Algorithms

Construction of the IP Virtual Server table takes place according to the scheduling algorithm used for the particular Virtual Server. To qualify for a highest litheness in the form of services we can cluster, Linux Virtual Server supports bellow mentioned scheduling algorithms.

1.6.1 Round-Robin Scheduling

Round-Robin scheduling assigns every clients request one after the other to the group of real servers. Through this algorithm every server gets job equally without considering work load and ability. Its similar to DNS round-robin, but its more detailed because it works on network level. It simply transfers clients requests to next server in list without considering total links and reply time of servers. Virtual Server Round-Robin scheduling is more efficient than conventional DNS round-Robin scheduling due to very efficient network based scheduling[22].

1.6.2 Weighted Round-Robin Scheduling

Weighted Round-Robin Scheduling assigns incoming request in sequence to the group of real servers however assigns more tasks to servers which have higher capability. If there is considerable dissimilarity in the capability of real servers group, then Weighted Round-robin scheduling is the best solution. The weighted round-robin plan aims to better cope with the different servers handling capacity. A weight is assigned to every server; its an integer value, which shows servers processing capability[22].

The new incoming request will be sent to server containing higher wait. During the construction of a weighted round-robin scheduling, scheduling series are created considering the server weights every time after rules of Virtual Servers are updated. Network Links are established to all real servers through the scheduling number series. Weighted Round-Robin Scheduling is efficient when servers capabilities are different in the real server group, but with high requests load this can create problems in equally assigning work load to real servers. For example it is possible that a large number of requests, requesting a huge reply can be assigns to one same server.

1.6.3 Least-Connection Scheduling

Least connection scheduling assigns incoming request to the real server which has less established connections. This scheduling algorithm carried all live connections entries in the IP virtual server table. Its a kind of flexible scheduling algorithm, in which there is an enhanced alternative when there is a high amount of deviation in the request burden. It is most excellent for a real server group where each server has approximately the same capability and performance characteristics, as Virtual Server will send requests directly to the real server with the smallest alive links. When a group of servers contains divergent capacity, then weighted least-connection scheduling is the best option[22].

At earliest in a quick look it may look like that the least connection scheduling can also be good, even if the server has different ways to treat with request and have different power, as the quicker server receive additional network links. But infect it will not carry out well in long run and the least-connection scheduling will fail to balance work load equally on all servers in the pool, when servers have different computational power.

1.6.4 Weighted Least-Connection Scheduling

Weighted least-connection scheduling is a supper form of least connection scheduling, the difference is that here we can allocate some kind of weight for every real server of the group based on their capabilities, so in this way servers which will be given higher wait will get a bigger share of active links to virtual server. The administrators can allocate weight for every real server and network links are allocated to every server, where the proportion of the present number of links for each server is proportional to its weight[22].

1.7 Tools for performance measurement of web services

There are many free open source tools available for the performance measurement of web services. Some of them are described below.

1.7.1 Apache JMeter

Is a java based application used for measuring different features of web services. Apache JMeter Is very flexible tool and used to analyze capability together on FTP-servers, queries, databases, Perl scripts, files and Servlets etc [9]. It produces intense work load on the server to check its power or to examine generally capability using diverse type of work loads. Useful graphical results can be generated to examine capability of server using high simultaneous work load. It can be used to simulate load and perform analyses and performance test of various type of servers like Web, HTTP, HTTPS, SOAP, Database, LDAP, JMS and Mail-POP3.

1.7.2 httperf

Httperf is another tool used for the measurement of web server capability and performance. It uses one of its useful services to generate heavy HTTP traffic and for

analysing and measurement of server capability and performance. The three main features of httperf are its strength, which provides the facility to produce and sustain server burden, SSL and HTTP protocols support and flexibility [10].

1.7.3 OpenWebLoad

OpenWebLoad is used for web applications work load performance analyses and test. This tool is very simple in use and it provides capability and performance analyses of the web applications in much realistic and professional way. OpenWebLoad is very helpful and constructive while performing effective enhancement, as one can see effect of enhancement without any delay after applying changes [11].

1.7.4 Pylot

Pylot is an open source tool used to perform test for measuring capability, performance and expandability of web services. HTTP work load test is used in Pylot, which are useful for capability judgment, investigation, and system analyses and correction. Pylot creates parallel, confirms server reply, and generate reports with useful metrics. It provides facility to perform test setup and examine results from GUI. It uses XML based test cases for send request whereas response from server can be examined through HTTP status values. Important features of Pylot tool are multi-threaded load producer, support for HTTP and HTTPS(SSL), graphical result report, GUI as well as console and shell mode [12].

Chapter 2

Linux Virtual Server

2.1 Linux Virtual Server Basic Terminology

Linux Virtual Server uses layer 4 switching for equally distributing clients request to different servers running those services. In this way Linux Virtual Server serves as Load Balancer. It has tremendously high-speed and makes such services to be available and accessible for thousands of concurrent requests. In this section we will discuss some of the LVS terminology which are most commonly used while setting up Linux Virtual server.

2.1.1 Load Balancer or Linux Director

Its the Node or Host that must have Linux with LVS Installed on it. This is the main Node, which is responsible for redirecting of packets. It receives packets from clients redirects them towards a real server in the server pool. As the service requested from clients may be provided from more then one real servers, LVS director basically servers as a special router, which transfers packets between client and one of the real server which basically provide that service [4].

Upon receiving clients request packets for a particular virtual service, decision making in the Linux director for selecting a real server is done by the scheduling method used in LVS. Only for the first packet this decision will be done by the scheduling algorithm, while all related incoming packets from the same client will be directed to the same real server. For virtual services connections integrity is very important. LVS director ensures this integrity by marking a service continued for a period of time. Through this it makes easier for LVS director to transfer all later incoming packets from a particular client to the same real server chosen by the scheduling method till a time-out expires. Data integrity is very important for some services, like FTP services. For such type of services, its very important that client's request must always be forwarded to same real real server [4].

2.1.2 End User or Client

End User or Client is the node that sends request for a particular service to the director. End User doesn't know any thing about real servers. It always sends requests to the director and assumes as it's requested service is provided by the director.

2.1.3 Real Server

This is the host that actually runs one or more services. Real Server is basically the final destination for a request made bye the client or end user for a particular virtual service. All real servers actually running some applications, which are responsible for processing clients request. Services offered by all real servers recognized by an IP address, port and also some time by a weight, if given by the scheduling method. Basically virtual service and real servers must have same port [4].

2.1.4 Virtual IP Address

All virtual services assigned an IP address through which Director handles request for that virtual service. These IP addresses are called Virtual IP addresses.

2.1.5 Real IP Address

IP addresses that are assigned to all real servers in the server pool.

2.1.6 Layer 4 Switch

layer 4 switch combines several connections on a shared medium. These connections are combined by multiplexer. It multiplex all TCP/IP connections and UDP/IP datagrams and transfer towards real servers. Linux Director discussed above receives all packets and selects a real server through the scheduling algorithm to forward all incoming packets.

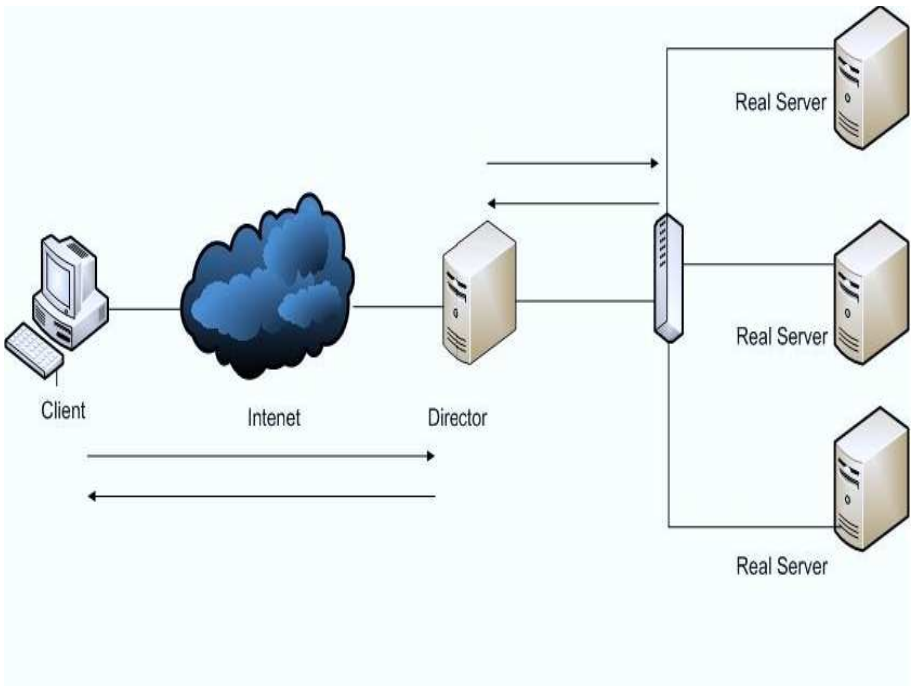


Figure 2.1: Virtual Server

Switch keeps in existence a small number of state for every end-user request connection managed by the cluster. This basically decides one real server from the cluster nodes to transfer packets. Switch can also keeps in existence work load on every real server of the cluster based on the total number of client's connections established to a particular real server.

By looking into TCP syc packet, switch determines a real serve in the pool, which is least loaded and transfers packets to that server. After that all following packets from the same client are transferred to the same real server, until a message is not sent to switch for instruction to transfer packets to another real node. When connection ends, a close message used to send to switch by one of the real servers which actually handles the request. Upon receiving this message switch closes connection state [17].

2.2 Virtual Server

In this section we will discuss what is virtual server, why we need virtual server, how virtual server works and methods through which LVS can be established.

2.2.1 What is Virtual Server

Virtual Server is a very expandable, variable and reliable server comes together with a set of real servers. It has the ability to equally distribute load on real servers through a Load balancer working on the Linux operating system [4]. The system architecture of the Linux Virtual Serve Cluster is very clear for the clients. Clients connects to the Linux Virtual Cluster system assuming if it were a single server handling all client's requests. Figure 2.2 describes a Virtual Server. Figure 2.2 shows that Linux Load balancer is connected to a number of real servers which constitute a cluster. This load balancer can be connected to the server cluster either through LAN or WAN. Load Balancer receives requests from the clients and forwards them towards a server in the cluster. This makes possible for all the services of cluster to be look like a virtual service on one IP address. Request dispatching uses different technique for load balancing. This system is very flexible and expandable, as nodes in the cluster can be easily removed for maintenance and added again without any service failure [18].

2.2.2 Why we need virtual server

As we know that Internet is rapidly growing and playing an important role in humans life. Its becoming necessity of every human being so the network traffic increases over the internet very fast. Due to this increase in the network traffic, server becomes overloaded very soon, as a result of heavy work load. This problem occurs very often for servers which are running some very frequently used and accessed services. To resolve this work load issue on servers, there are some solutions one of them is to keep using single server through updating, to meet high performance requirements.

It has been seen that this solution doesn't work for longer time. Soon a more higher performance server will be required, as network traffic is continuously increasing day by day. But upgrading servers frequently is not easy and also its very costly solution. Second solution is to use multiple servers as a cluster for such services which are very

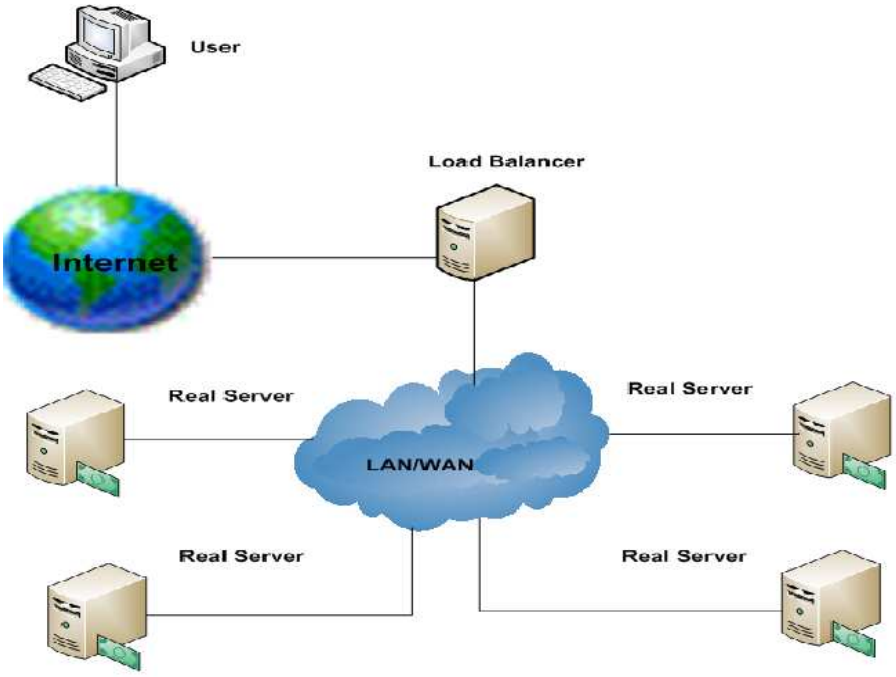


Figure 2.2: Virtual Server

popular over the internet and causes heavy work load on the servers. This is very flexible solution, as it allows to add number of servers in the cluster according to the increase work load on the cluster. So its an easily expandable, very efficient and cost-effective solution to build a cluster of servers for such network services which generate heavy network traffic over the internet and increases work load on the server [18].

2.3 Methods for Constructing Cluster of Servers

We will discuss two methods used for constructing server cluster.

2.3.1 Load Balancing Cluster through DNS

DNS based load balancing cluster is one of the simplest way to construct the cluster. It simple uses Domain Name System for sending requests amongst all the servers in the cluster, by obtaining IP addresses of servers against Domain Name. Upon arriving a DNS request to the Server, Server picks one of the IP addresses of the Server Cluster. This is done through the scheduling algorithm used. All the following incoming requests are sent to the same sever in the cluster until a time-to live value is not expired. In Load balancing cluster through DNS , it's not an easy job for a server to work properly in high load state, because of client's cache nature which causes imbalance load between the Servers. Time to live (TTL)value of the mapping must be carefully selected on the Server. For very small TTL value DNS traffic will be higher where as the server performance capacity is a bottleneck in this case, where as for very high TTL value Load Balance will be even too bad. As different users generate different

traffic over the internet, some people download a lot of data while others may only view a single page. In this case even if the TTL value is placed to zero, the different load creating behaviour of clients can lead to Load imbalance. Additionally it's not trustful solution when one of the server goes down in the cluster, clients will still use the failed server mapping IP address [18].

2.3.2 Load Balancing Cluster through Dispatcher

Dispatcher are also called Load Balancer and a dispatcher can be used for distribution of work load between multiple servers in a group of servers or cluster. This makes possible for the equivalent services of the servers to be used as virtual server service on a single IP address. End user interaction assumes just like it is interacting to only one server and they doesn't know about other servers or cluster of servers. Comparison of DNS based Load Balancing and Dispatcher based Load Balancing shows that Dispatcher based load balancing distributes load equally on each connection as a result it provides effective load balancing between the servers. This is very flexible solution, management of servers becomes much more easier, as its possible to add or remove a server from the cluster for maintenance and this will not cause service interruption for the clients.

There are two level of Load Balancing that can be provided, one is called Application level Load balancing and other other is called IP level Load Balancing. Examples of application level Load Balancing are reverse-proxy and pWEB. These basically uses application level Load balancing algorithm to create a scalable web server. What they do, actually they transfer client's HTP request to some another server in the cluster, results comes back and then this result is transferred back to the original client. Application level Load balancing is also a bottleneck as number of servers increase to five or little bit more, depending on the capacity of each server. Where as IP level Load balancing is suitable, as it has less overhead and it can use number of servers up to 100 [18].

2.4 Packet Forwarding Methods for LVS

Linux virtual servers transfer packets from client to real servers and back from real servers to client in three different ways; Network Address Translation (NAT), IP - IP encapsulation (tunnelling), and through direct routing all exist in the Linux Load balancer or Director [4]. All packet forwarding methods will de discussed one by one in detail.

2.4.1 Virtual Server Through NAT

Network Address Translation is a method that manipulates source and destination's IP addresses and ports of the packets. One of its most frequent and efficient use is IP masquerade, IP Masquerade makes possible for the private network to connect to the outward for internet access [4]. Looking for Layer 4 switching point of view, packets coming from the clients are manipulated in such a way that the destination IP address and Port address are replaced with one of the server's IP address and Port address in the cluster selected by the scheduling method used. In NAT server sends reply packets

back to the Linux Director and mapping is removed before sending to the client, so that end user gets reply back from expected node.

LVS through NAT is one of the simplest way to implement Linux Virtual Server test, as it doesn't require any alteration in the configuration of the real servers. Besides the others two methods used in Linux Virtual Server LVS Direct Routing and LVS Tunnelling, LVS NAT only requires that all real servers must have TCP/IP stack working. It mean none of the configuration is required at server level except to set-up their rout tables [19].

One of the main advantage of NAT Virtual Server is that all real servers can be used in a private network, as client's need only one IP address of the Linux Director and also real servers are free to use any Operating System having TCP/IP protocol support. On the other hand NAT is not highly scalable, which is one the disadvantage using NAT Linux Virtual Server. The bottleneck in NAT Virtual Server is the Director or Load Balancer used. This is because Load Balancer has to rewrite both packets, client's request packets and server's replies packets too. As a result it can support up to maximum of about more or less 20 servers [18].

Though the Load Balancer is the Bottleneck for NAT Virtual Server, it still fulfils the performance requirements of many servers. There are two ways to overcome or solve this problem, one method is to use hybrid mechanism, through the DNS Hybrid solution many Linux Director or Load Balancer are used together with their own real servers. All the Load Balancers used to be grouped into a single domain name through Round-Round DNS [18]. Other method is to either use IP Tunnelling or direct Routing Virtual Server.

The more and more increase in the usage of the internet IP addresses, leads to the lack of IP addresses and also creates a number of security threats. To overcome these problems many networks uses private IP addresses. These private IP addresses can't be used over the internet, so one can't be connected to internet having private IP address. Private Network's need to connect to the internet and this is fulfilled by Network Address Translation. Network Address Translation implements mapping between different private network and also to the internet. There are different type of mapping, one is called Static NAT in which mapping is done one to one where as in dynamic NAT mapping is done many to one. Linux IP masquerading works through N to one mapping where TCP/UDP port translation is also done together with IP address translation. Virtual Server through Network Address Translation (NAT) is implemented through Network Address Port Translation [18].

Working of Linux Virtual Server Through NAT picture

To understand the working of Linux Virtual Server through NAT, consider the following Figure 2.3. Some number of real servers are connected through a switch to the director or Load Balancer and the Load Balancer is connected to the internet. Any user on the internet can access the directors IP. When clients accesses a service from the server cluster, the request for that service goes to the Director which is the only IP address available for external users to connect to the server cluster. When packets are received by the Load Balancer, then it's responsibility is to check destination address and port number of the incoming packets. If it finds a match for the service according to the table entries in the virtual server, then it selects an actual real server from the

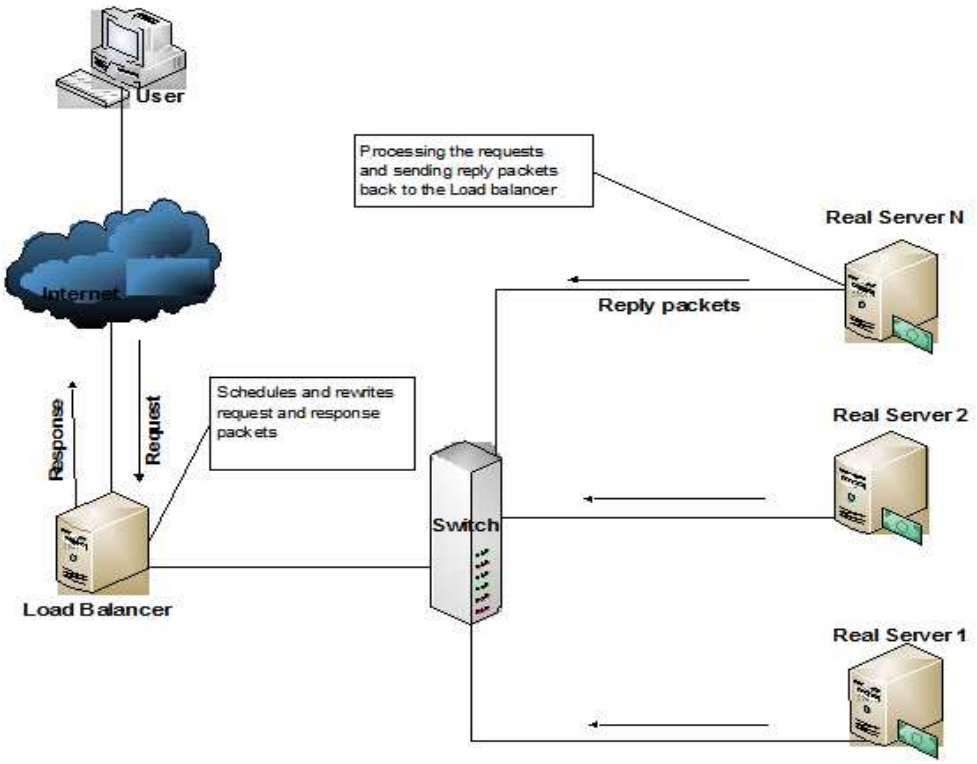


Figure 2.3: Linux Virtual Server through NAT

cluster according to the scheduling method used at the Load balancer.

There is a table in the Director called hash table. This table is used by the Director every time a new connection is established, to maintain a record of the established connections. After this director has to rewrite the IP address and Port number of the destination node to one of the server which is selected by the scheduling algorithm and then it has to send the packet towards that real servers, which has to handle the request. For all the following incoming packets for this established connection, the real server will be found through the hash table entry and destination IP address and port will be replaced to the IP address and port of the real server found through hash table. In this way all packets of a particular request will arrive to the same real server.

When the real server receives the request, then it has to handle and process the request and send reply back to the client. In Linux Virtual Server through NAT real server doesn't know destination address of the client so it will send reply back to the Director, which has to rewrite the source IP address and Port number of the packets to the virtual service. Through this it appears for the client, that the reply arrive from the virtual service address which is the Director or Load Balancer. Entry from hash table of a connection will only be erased, when time-out expires or due to any other reason connection breaks.

2.4.2 Linux Virtual Server through IP Tunneling

As discussed in the above Linux Virtual Server through NAT, that both request and response packets must pass through the Load Balancer or Director and we concluded that Load balancer can be bottleneck in LVS through NAT, when number of real servers increases from 20 as all the network interfaces ultimately have a maximum throughput limit. Usually for all services like web services, the requesting packets from the clients are much shorter whereas the reply packets from the real servers are much bigger due to large data.

The Load Balancer's limitation can be resolved using the Linux Virtual Server IP Tuneling, as in Virtual Server through IP Tunneling the Load balancer's only job is to transfer packets from client to different real servers in the server pool and it's the real server's responsibility to send the reply packets back to the actual client directly. Removing this extra burden from the Load balancer enables the Load balancer to schedule large number of requests. This makes possible for the Director to handle up to 100 requests or even more without any Load Balancer's bottleneck issue. Therefore using IP tunnel one can effectively increases the maximum number of real servers supported by a Load balancer. The throughput of the Virtual Server can greatly be increased, that is using Virtual Server through Tunnel one can easily increase throughput of Virtual Server up to 1Gbps, though the Load balancer link is limited to 100Mbps [18].

These special functions of Virtual Server through IP tunneling helps to establish a high performance Linux Virtual Server. It is very good to establish virtual proxy server, as proxy server after getting client's request has ability to access internet to get objects and then it can send beck to the actual client directly. But its important for all the real servers to have IP tunnel protocol support [18].

Linux Virtual Server Tunnel is basically based on Virtual Server Direct Routing (LVS-DR). Both have the same advantages of increase in throughput as well as reliability and scalability. Linux Virtual Server Tunneling can only be used with such real servers which have implemented IP encapsulation. Load balancer upon receiving requesting packet from the client, encapsulate it within an IPIP packet and then transfers it to one of the chosen real server. Only if real server has implemented IP encapsulating, then it can decapsulate this packet. So real servers must have an operating system that must support IP encapsulating protocol, in the start only Linux has been supporting this protocol but then FreeBSD and W2K also started to support it [19].

Linux Virtual Server through IP Tunneling is very flexible that there are no restriction for the real server as in the Virtual Server through Direct Routing. All real servers and Load Balancer can be on two different remote networks. This allows for the real servers to be in another country. If Director and real servers both are on the same network, then there is no need to use Linux Virtual Server Tunneling approach. In this case Linux Virtual Server-Direct Routing is the best approach [19].

Working of Linux Virtual Server Through Tunneling

IP Tunneling is actually an encapsulating technique to put IP datagram within another IP datagram so that packets sent for one destination can be wrapped and transferred to a new destination IP. This LVS-Tunneling bas been greatly used in many networks especially in Mobile-IP, tunneled network and IP-Multicast [18].

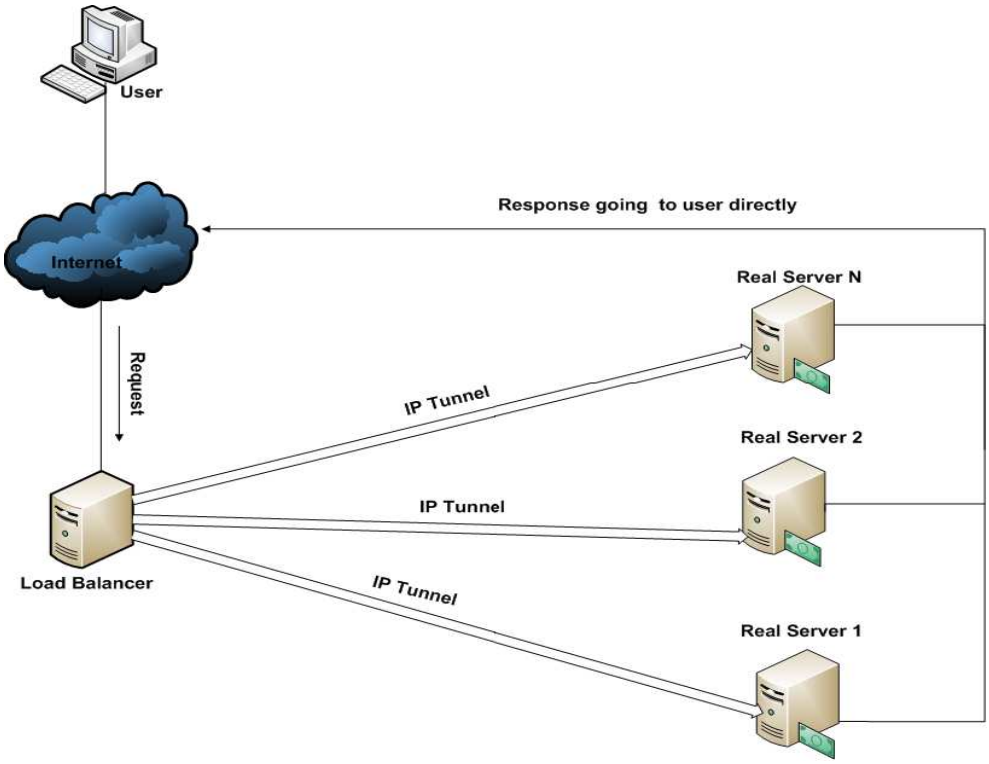


Figure 2.4: Linux Virtual Server through IP Tunneling

To understand how Linux virtual Server works through IP-Tunneling consider the Figure 2.4. The difference from the Linux Virtual Server through NAT to that of the Linux virtual Server through IP-Tunneling is that Director sends the client's requesting packet towards the real servers through IP-Tunneling whereas the real servers instead of sending reply packets back to the Director, send reply packets directly to the actual client through Network Address Translation.

When a client sends request to access a virtual service from the Linux Virtual Server implemented through IP-Tunneling, first of all packets with destination address as virtual IP address, i e the address of the virtual server, reaches to the Director. The Director finds the packet's original destined IP and port. Through matching the IP and port it fiends, either it is a request for the virtual service. If so then it selects one of the real server from the server pool through the scheduling method and then after adding this connection entry in the hash table, the Director has to encapsulate incoming packet into another IP datagram and then transfers this packet towards one of the selected real server. All the following incoming packets which will match to this connection and the selected one of the real server from the hash table, will again be processed to encapsulate and transferred towards the real server selected.

As the real server will receive encapsulated packets it has to decapsulate the packets, so all the real servers must have the encapsulated protocol enabled. After processing the request the real server will find the destination address table will be removed.

The real servers in LVS-IP-Tuneling needs to have a real IP address but this real server can be on any network and will transfer the reply packet directly towards the

client. When a connection breaks or its time-out expires, then its entry from the hash table will be removed. The only limitation is that all real servers need to be support encapsulating. To accurately decapsulate the packets, real servers tunnel devices need to be configured [18]. After decapsulating the packets server can know, that this packet is sent to me. Then real server does requested processing and transfers the reply back to client.

2.4.3 Linux Virtual Server through Direct Routing

Linux Virtual Server through Direct Routing is also a Load Balancing technique, that can be implemented in the Linux Virtual Server. It transfers requesting packets directly to the one of the real servers in the server pool through replacing MAC address of the packet with the MAC address of the chosen one of the real server through the scheduling algorithm used. It has been said that due to the very low overhead of replacing MAC address, it provides high scalability and performance as compared to the other LVS-NAT and LVS-IP-Tunneling [20].

Implementing Linux Virtual Server through Direct Routing provides huge increase in performance when it's performance examined comparing with the other Linux Virtual Server network topologies. Direct Routing as by name shows that, it makes possible for the real servers in the Linux Virtual Server cluster to transfer reply packets after handling the request directly to the original requesting client without sending back to the director,hence reducing work load on the Director which then decreases the performance problems of LVS. To fulfil this Direct Routing needs that the Director and the real servers must be connected to one physical network and also all real servers must have the ability to transfer reply packets directly to the end user [8].

Comparing LVS-DR with Linux Virtual Server IP-Tuneling shows that, one another benefits of LVS-DR is that it doesn't have tunnel overload too. The only overhead of the LVS-DR is that, it requires both real server's and one of the Director's interface to be on single network.

Virtual Server through Direct Routing shares virtual IP address between the Director and the real servers. One of the director's interface is also configured with the virtual IP address, that accepts the client's requesting packets. As mentioned above then it transfers packet directly towards one of the selected real server. Real servers in the server pool contains one of their interface to be configured as the virtual IP address otherwise Director transfers packets to the local socket. This makes possible for the real servers to transfer reply packets directly to the original client after processing the request locally [18]. As mentioned early the Director as well as all the real servers need to have one of the interface directly linked through the Switch or Hub. To understand how Linux Virtual Server through Direct Routing works consider Figure 2.5.

When a client sends request to access a virtual service from the Linux Virtual Server implemented through Direct Routing, first of all requesting packets with destination address as virtual IP address, i e the address of the virtual server, reaches to the Director. The Director finds the packet's original destined IP and port. Through matching the IP and port it fiends, either it is a request for the virtual service. If so then it selects one of the real server from the server pool through the scheduling method and then after adding this connection entry in the hash table, the Director directly transfers this packet towards one of the selected real server. All the following incoming packets

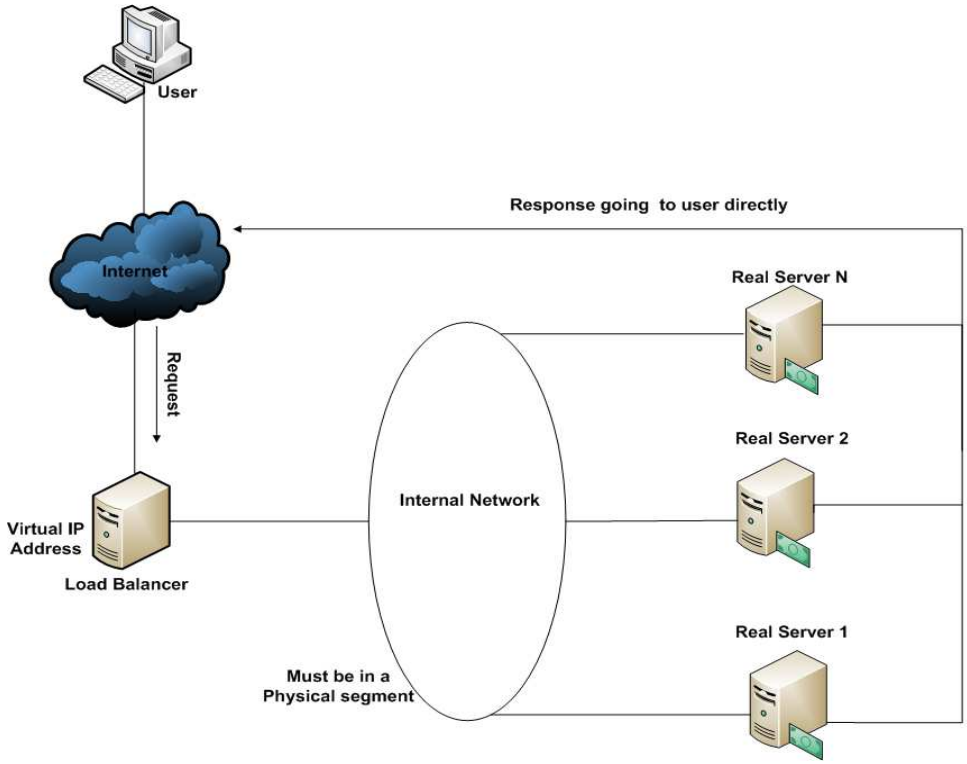


Figure 2.5: Linux Virtual Server through Direct Routing

which will match to this connection and the selected one of the real server from the hash table, will again be directly transferred towards the real server selected.

When a connection breaks or its time-out expires, then its entry from the hash table be removed. The only requirement is that the Direct Routing needs that the Director and the real servers must be on the same physical network and also all real servers must have the ability to transfer reply packets directly to the original client.

The director's only job is to replace the MAC address of the requesting packet to the MAC address of one of the selected real servers and to transfer the packet over the LAN. That is why the Director and the real servers needs to be connected directly through a single Local Area Network segment (LAN)[18].

Chapter 3

Linux Virtual Server Experimental Setup

3.1 Experimental setup

The experimental setup consist of a Linux Virtual Server Load Balancer, Scheduler, packets rewriter or its also called Director. Director is connected to a number of real servers through a switch. This director is responsible for receiving all incoming requests from users and directing these requests to one of the real servers in the cluster which is also called IP-level load balancing of the traffic. Director is the only single point of interaction between user and the real servers in the cluster. It makes possible for the director to present services from all real servers as a single virtual service on a virtual IP address.

When user will send http request for data, first this request comes to the director. Director then redirects this request to one of the real server in the pool according to the scheduling algorithm used by the Director. All servers upon processing all coming request from the director, send reply back to the Director and its Directors responsibility to send reply back to original requesting user. At the end client doesnt know anything about real servers, because for the client replies comes back from Director or Load Balancer.

Load Balancer or Director uses two network interfaces, one which connects Load Balancer to the Internet, so that clients can access Virtual Server and one which connects Load balancer to the Local Area Network (LAN). As all real servers are connected to this LAN, so adding or removing a real server to the LAN is very easy which makes the setup design very scalable.

All scheduling algorithms will be used one by one with varying number of real servers in the cluster, to conclude which scheduling algorithm suites best with certain number of real servers. As the internal link to FrontEnd Load Balancer is 100MB and external link is of 1GB, which is bottleneck for the Load balancing algorithms. There will be an upper limit for number of real servers for the Load Balancer with 100MB Link. After certain number of real servers, adding more real servers will not give more performance benefits due to director's bottleneck. This cluster is such a scalable cluster that if director receives an increase number of requests, the Load Balancer can easily manages by just adding some more real servers in the Server pool. One of the

main advantages of LVS is that it allows us to use any operating system running node that supports TCP/IP.

In this Experiment we will setup a scalable and load-balancer network servers cluster using Linux Virtual Server Project. There will be maximum 15 real servers in the pool all providing same network services, downloading web pages. Network traffic will be generated through running scripts in a loop on the user end which will download web pages from one of real server through Linux Virtual Server.We will generate high, medium and low traffic through downloading different size of web pages.

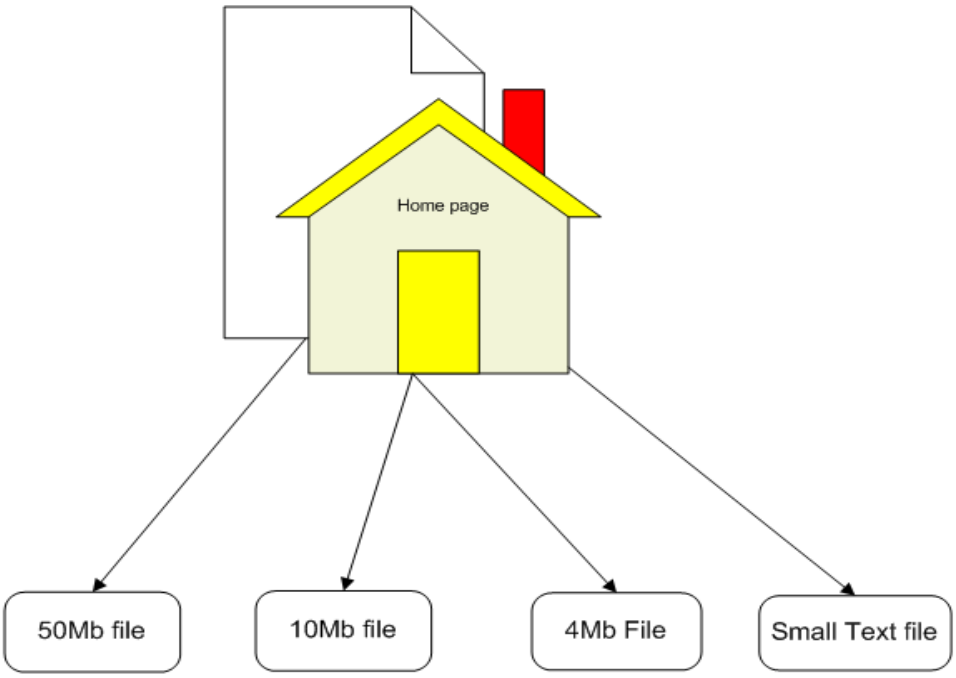


Figure 3.1: Downloaded Web pages

3.2 Experiments

we are using Linux Virtual Server kernel 2.7 which doesnt need any additional patching for setting up Linux Virtual server.

3.2.1 Installing IPTables and IPVsadm

We need to have IPTables and IPVsadm tools on our system for the configuration of Kernel to make it as Linux Virtual Server. IPTables is a useful tool which is used for building, mainting network address translation and packet filtering rules in the kernel [13]. IPTables are also used for enabling IP masquerading for all the real servers in the server pool.

One of the important tools that we will use for the administration of Linux Virtual Server is IPVsadm. With the help of this tool we will setup different forwarding methods and as well as impliment different scheduling methods.

The director transfers service request of the client to one of the real server in the Linux Virtual Server pool, which are responsible for performing actual service request. It supports two protocols TCP and UDP. Packets forwarding supported methods are NAT, tunneling and direct routing. Scheduling algorithm can be used from one of the following, round robin, weighted round robin, least-connection, weighted least-connection, locality-based least-connection, locality-based least-connection with replication, destination-hashing, and source-hashing [15].

After installing IPTables and IPVsadm we need to set IP forwarding on the server. This can be done by editing the sysctl.conf file and setting net.ipv4.ip_forward = 1. Then we need to start IPTables services which makes possible for the director or load Balancer to redirect response from the real servers to the actual client who requested the service. This can be done through the following command [13].
service iptables start
After having installed IPTables we are able to enable IP masquerading. Its a networking method used in Linux which makes possible for the internal LAN computers to connect to the Internet by connecting to the Linux Box which is connected to the external Internet. IP Masquerading is such a powerful method that makes possible for the machines which are on LAN to connect to the internet, even if LAN machines are not configured to assigned proper official IP addresses.
In this way masquerading allows Internal Machines to connect to the internet invisibly through the masquerading gateway. For the others machines on the internet, it will appear that all traffic comes from the gateway, which is directly connected to the internet. In addition to this facility, IP masquerading also provides basics for making network highly secured [16]. Front End Linux Virtual Server Director or Load Balancer is connected to Internet through Interface eth0 as well as its connected to the Local Area Network through Interface eth1, where all real servers are also connect. IP masquerading is done through the following commands.
iptables -t nat -P POSTROUTING DROP
This will setup the default policy or drop it. By dropping mean that, if there will come a packet that will not match to the policy rules, it will be dropped. Through this we are implementing security on the server for not masquerading every forwarded packet towards server.
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
This enables NAT and it enables masquerading of all real servers IP addresses to the Load Balancer Or director s external interface address, which connects it to the Internet [14].

3.3 Configuration of Linux Virtual Server

Now IPVsadm is ready and we can use it to configure Linux Virtual Server. Before starting LVS configuration we need to setup IP addresses for all the real servers as well as for Director. We need to put all real servers on a LAN and allocate them private IP addresses as well as Internal interface of the Director on the same subnet and this

interface of the director will be default gateway for all Internal LAN real servers. The interface that connects director to the Internet will be assigned a public IP address.

We are running Rocks cluster distribution 5.0 Centos Linux OS, Director's external link is 1GB where as internal link is only 100MB. All other nodes have 1GB links, motherboard has two cards on each machine one for 1GB and one for 100MB. Other specifications are given below
P Procurve Switch 2724
Intel serverboards SE7210TP1-E (forntend and nodes)
Built-in GB Ethernet cards.
Frontend uses also a 100Mb Card (built-in)
Dual core pentium 4 (2.8 GHz) (frontend) (4GB RAM)
Dual core pentium 4 (2.8 GHz) X 8 (compute0-0 - compute0-7) (2GB RAM)
Dual core pentium 4 (3.0 GHz) X 8 (compute0-8 - compure0-15) (2GB RAM)
SATA disks 80 GB on compute nodes
SATA disks 400 GB on frontend

3.3.1 Implementing LVS via NAT using Round Robin Scheduling

The experimental setup of LVS via NAT is shown in the Figure 3.2 below. Three tests will be conducted using Round Robin scheduling algorithm. Common configuration in all three tests will be following.
Packet forwarding method: -m –masquerading
Scheduling Algorithm: rr –round robin
We will vary the number of real servers of the Linux Virtual Server cluster in each test from 2 to 8 real servers and in all three tests we will place different size data files on the real servers for downloading.

Test Case A Method

In test case A we will use the following configuration togather with common configuration as mentioned above.
Number of real servers: 2, 4, 6 and 8
we will vary the number of real servers from 2 to 8 to check the performance of Linux Virtual Server with different number of real servers. setupLVS.sh script in Appendix is used to configure Linux Director for a given number of real servers to forward incoming requests from clients directed towards port 80 on 128.39.73.11, to port 80 on given number of real servers in the server pool. NAT or masquerading is used as packet forwarding method and Round Robin is used as scheduling algorithm. In this case we are using NAT or masquerading as forwarding method, so we need to set default gateway of all the real servers to the internal interface of the Director, which is connected to the LAN. The following command will set Directors internal interface as default route of the real servers.
echo "1" > /proc/sys/net/ipv4/ip_forward
Client.pl script in Appendix is used to download different web pages countinuesly in random manner. Four web pages of size 76Kb,103Kb, 505Kb and 1.8Mb are placed on real servers. Client.pl script is used to run on four differet clients to downloads these pages randomly to generate different size of traffic on the Load Balancer. Crontab is

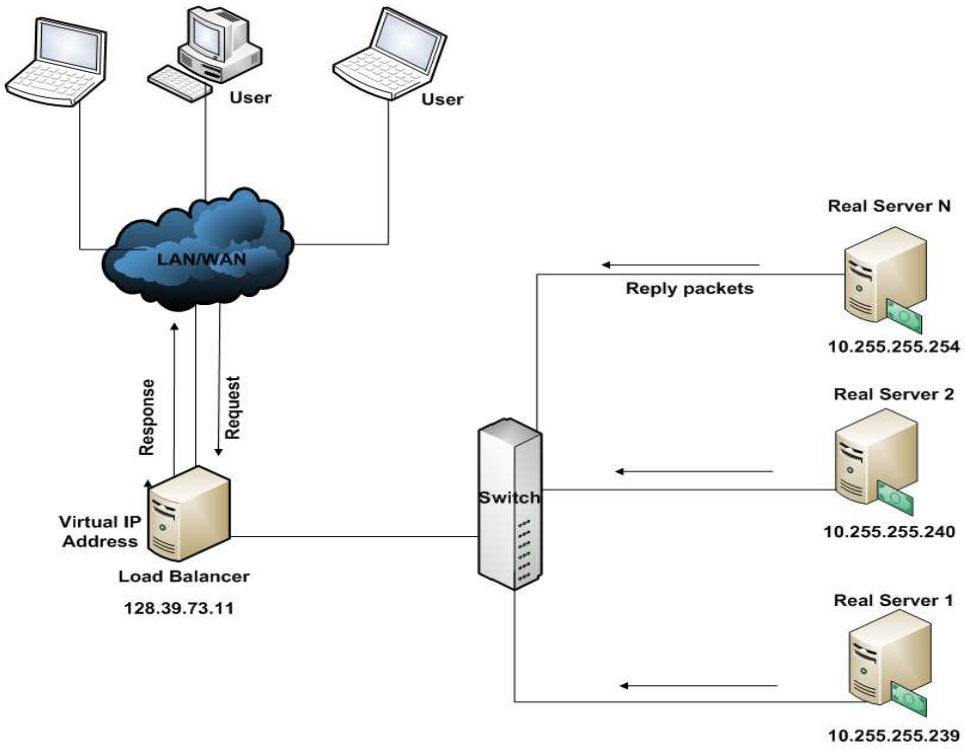


Figure 3.2: Experimental setup of LVS via NAT using Round Robin Scheduling

used to save ipvsadm throughput-stats, rate-stats and conection-stats. Crontab is set to save ipvsadm stats data after every minute to three different files. These files are used to analyse data and then to generate different graphs.

Test Case A Results

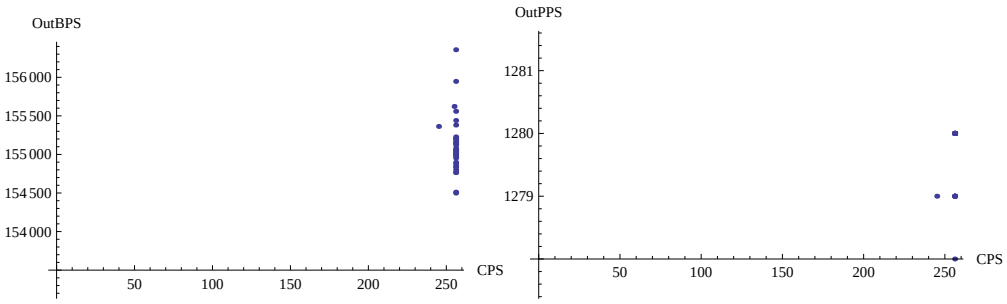
First i configured Linux Virtual Server with two real servers and i used four clients to run client script for downloading web pages. Crontab is used to save different stats data on director. . These files are mentioned below.

Rate-stats-2 to rate-stats-8. These files are used to store Input packets per second, output packets per second, input byte per second , output byte per second and connections per second to director. These files store stats data of Director as well as of the real servers used. But i am interrested in Director's stats data. Setup is used to run for 2 to 8 real servers and crontab stored rate stats respectively in four different files.

Throughput-stats-2 to throughput-stats-8: these files are used to collect total number on connections, input packets, output packets, input bytes and output bytes.

Conection-stats-2 to conection-stats-8: thes files stores total number of connections with clients IP address, Director IP address and IP address of the chosen real servser for that client's request.

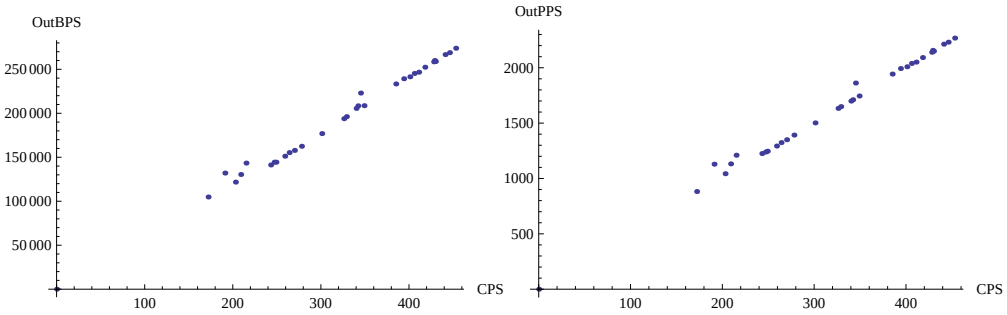
Important statistical data is extracted from these files for analysis and generating graphs. Mathematica 7 is used to generate different graphs. These graphs are given below.



(a) OutBPS (b) OutPPS

Figure 3.3: OutBPS and OutPPS with 2 real servers

Graph (a)OutBPS represents connections per second on X-axes and output bytes per second on Y-axes. These graphs are generated using Linux Virtual Server with 2 real servers. We can see that graph is linear and maximum number of connections per second are 250, maximum outBPS are 156000 and maximum outPPS are 1281. Graph showes that Linux Virtual Server is working fine.
Now i would like to see these graphs with 4, 6 and 8 real servers to find any difference. I checked these graphs wiht 4, 6 and 8 real servers. In all cases graph grows linearly and abviously with increase number of real servers, the result was an increase in connections per second as well as increase in outBPS and outPPS. All graphs can't be shown here, but for reference below Figure 3.4 shows graph of result with 8 real servers.

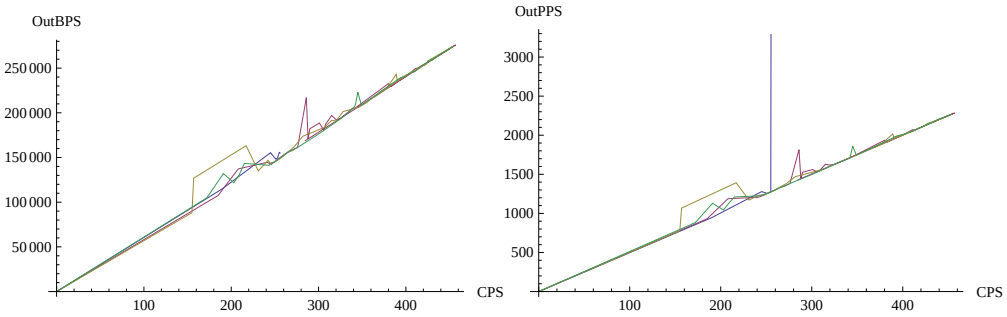


(a) OutBPS (b) OutPPS

Figure 3.4: OutBPS and OutPPS with 8 real servers

You see that with increase in real servers, total number of conections per second handled by the Director is increased similarly outBBS and outPPS also increasing respectively. This shows that there is no problem for the Director to handle more connectiins with increase in real servers. This is because files size is very small which are used to download from clients to generate trafic on the Director. So such small files does't create much load and Director can easily manage to handle new connections.

Figure 3.5 shows graphs of average throughput. Mathematica commands used to generate average graphs is given below. Average graph is used for all 2, 4, 6 and 8 real servers setup. All average graphs shows that when we increase number of real servers,



(a) OutBPS (b) OutPPS

Figure 3.5: Average OutBPS and OutPPS with 8 real servers

number of connections per second increases and as well as outPPS, outBPS, inPPS and inBPS increases.which shows that Linux Virtual Server works fine with the given load of the downloaded files. As we increase real servers Director disributes request to more servers, which increases throughput as a result.

```
avr[x_List] := Table[Apply[Plus, x[[i]]]/Length[x[[i]]],
               {i, 1, Length[x]}]
avrdata[x_List] := Sort[avr[Sort[GatherBy[x, First]]]];
avrcpsOutBPS2 = avrdata[cpsOutBPS2];
avrcpsOutBPS4 = avrdata[cpsOutBPS4];
avrcpsOutBPS6 = avrdata[cpsOutBPS6];
avrcpsOutBPS8 = avrdata[cpsOutBPS8];
avrcpsOutPPS2 = avrdata[cpsOutPPS2];
avrcpsOutPPS4 = avrdata[cpsOutPPS4];
avrcpsOutPPS6 = avrdata[cpsOutPPS6];
avrcpsOutPPS8 = avrdata[cpsOutPPS8];
ListPlot[{avrcpsOutBPS2, avrcpsOutBPS4, avrcpsOutBPS6
        , avrcpsOutBPS8},PlotJoined -> True ,
         AxesLabel -> {"CPS", "OutBPS"}]
ListPlot[{avrcpsOutPPS2, avrcpsOutPPS4, avrcpsOutPPS6,
         avrcpsOutPPS8}, PlotJoined -> True,
         AxesLabel -> {"CPS", "OutPPS"}]
```

Graph (b)outPPS shows suddenly very high outPPS, due to the result of outPPS using 2 real servers. I don't know why but suddenly outPPS jumped from 1279 to 91866 and then immidiatly came back to 1279.

Test Case B Method

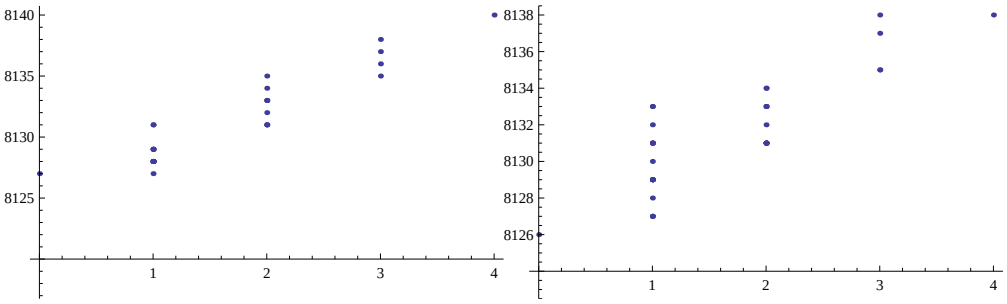
Test case B is designed to generate such traffic on the Director, that causes heavy load on the Director, to check the maximum throughput of the Linux Virtual Server. In Test Case A, downloaded files size was very small and there was no problem for the Director to handle incomming client's request, increase in real servers causes an increase in total number of connections per second and also out packets per second.

To overcome this, we needed heavy traffic on the Director ,to know whats the maximum throughput of the director. Keeping in mind that Link to director is 100MB, which is bottleneck for the Director's throughput. So the maximum throughput will be for 100MB Link of the Director's to the outword. For generating heavy load on the Director we have placed the following files on the real servers.
File0 = 41Kb.
File1 = 4Mb.
File2 = 10Mb.
File3 = 50Mb.

I hope that downloading these files will generate enough traffic Load on the Director to determine the maximum throughput of the Linux Virtual Server director. All other configuration will remain same as in Test Case A. We will vary the number of real servers to check any difference in throughput with different number of real servers in the linux Virtual Server. SetupLVS.sh script will be used to setup Linux Virtual Server with given number of real servers. Same four clients will be used to run client.pl for atleast 30 minutes. Crontab will be set to save new throughput result in a different location for all 2, 4, 6 and 8 real servers setup.

Test Case B Results

The only difference in test case A and B is the size of files placed on the real servers for download. Same way as in Test Case A, Linux Virtual Server is configured with setupLVS.sh script using Round Robin as Scheduling algorithm and then used this test for 2 to 8 real servers separately. Four clients are used for sending request continuously by running client.pl script. Crontab is used for storing stats results into different files for different number of real server's tests. Status data is analyzed using Mathematica and the following graphs are produced.

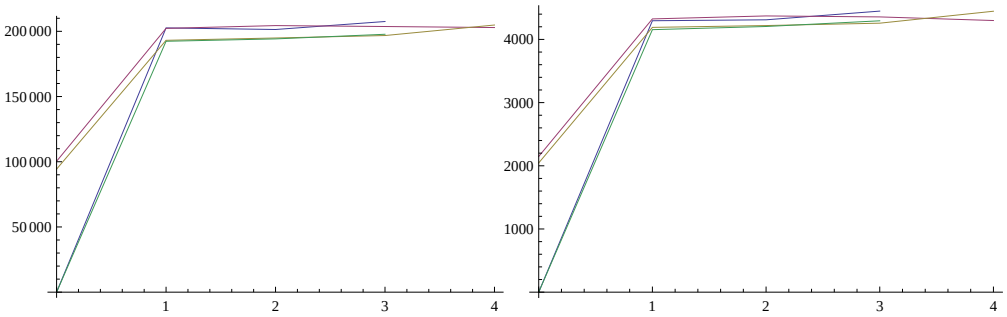


(a) OutPPS4 (b) OutPPS6

Figure 3.6: OutPPS with 4 and 6 real servers

The above OutPPS4 is actual graph representing number of connections per second to total number of output packets per second with 4 real servers and OutPPS6 is the actual graph representing number of connections per second to total number of output packets per second with 6 real servers. It has been seen that with 2 real servers graph grows linearly up to around 8140 OutPPS but after that its not going up. The above two graphs shows OutPPS results with 4 and 6 real servers, but you can see that there

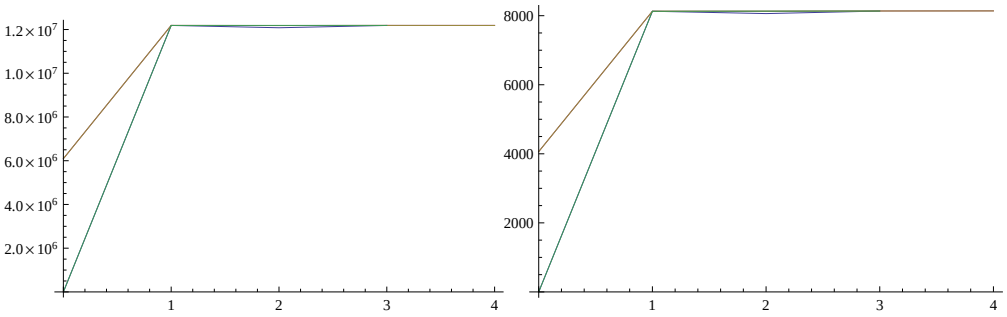
is no difference in throughput with 4 and 6 real servers and same result is with 8 real servers.



(a) InBPS (b) InPPS

Figure 3.7: Average InBPS and InPPS with 2-8 real servers

Numbers of graphs are generated. As it's not possible to present and discuss all tests results. So I decided to present only average graphs. Above two graphs, shows average InBPS and average InPPS. (a) InBPS shows CPS to InBPS and (b) InPPS shows CPS to InPPS for 4 tests from real server 2 to 8.
All graphs shows almost same result. All graphs goes up to a common point and the goes straight forward. This shows that Linux Virtual Server reached its maximum possible throughput and it doesn't make any difference by adding more and more real servers to the cluster.



(a) OutBPS (b) OutPPS

Figure 3.8: Average OutBPS and OutPPS with 2-8 real servers

The above two graphs shows an average result of OutBPS and OutPPS to CPS of all 4 tests using varying number of real servers. The maximum OutBPS reached is 1.2x10$\hat{7}$ which is the maximum throughput that can be achieved having 100MB link. These graphs shows that downloading 10Mb to 50Mb files over 100MB link through the Director causes Linux Virtual Server to reach the Director's bottleneck. Downloading these files generates heavy traffic on the Director and it's not possible for Director to handle more connections, to increase CPS and output rate. This result has been very useful to determine the maximum throughput over 100MB link.

Test Case C Method

In Test Case A downloaded data files size was very small that's why we did not see any difference in throughput in four different test using number of real servers from 2 to 8 with scheduling algorithm Round Robin. Then in Test Case B, I replaced those small data files with very higher data files that were up to 50 Mb files. It has been seen that generating traffic for downloading such big files from the Linux Virtual Server causes Director to quickly reach the bottleneck. Test Case C is designed to have smaller and medium data files on the all servers. These data files are from 100K to 10M sized. Data files used for downloading are given below.

100K = file0	400K = file3	800K = file6
200K = file1	500K = file4	1.0M = file7
300K = file2	600K = file5	10M = file9

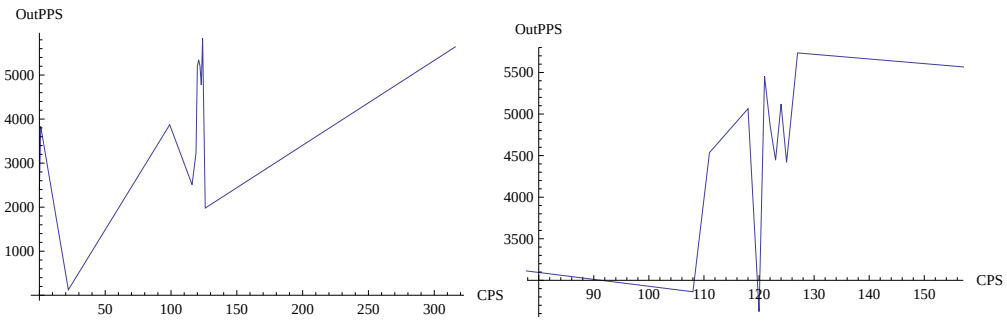
These data files collectively can be assumed as a small website having some small text and pictures and one or two audio files of up to 10M.

Test Case C Results

Linux Virtual Server is configured using setupLVS.sh script shown in Appendix. Test C is also used to run four times to collect throughput data of Linux Virtual Server with real servers from 2 to 8. The same scheduling algorithm Round Robin is used in Test Case C to compare throughput with Test case A and B.

Crontab is used to save throughput data of all Tests in different files. This data is analyzed and organized in a suitable way for Mathematica to real these throughput data files and generate different throughput grapgs.

Using Mathematics different Rate stats to connections per second graphs are generated. Then averaged data graphs are also generated. All of the graphs can't be shown, but some of the graphs are given below for references.

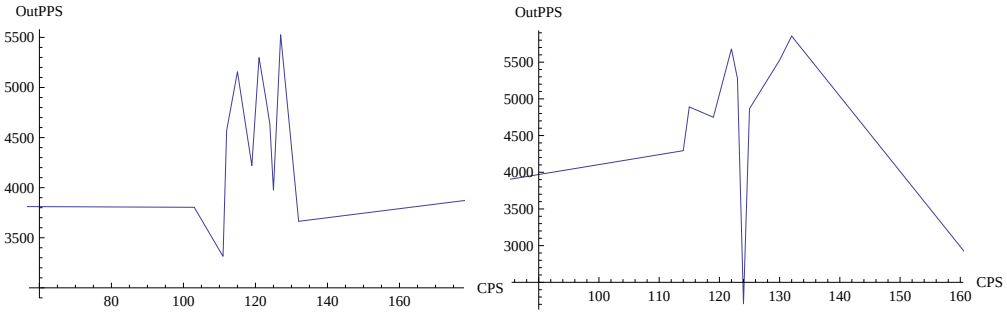


(a) OutPPS2 (b) OutPPS4

Figure 3.9: OutPPS with 2 and 4 real servers

Graph (a) shows connections per second to output packets per second implementing Linux Virtual Server with 2 real servers. It has been seen that with 2 real servers

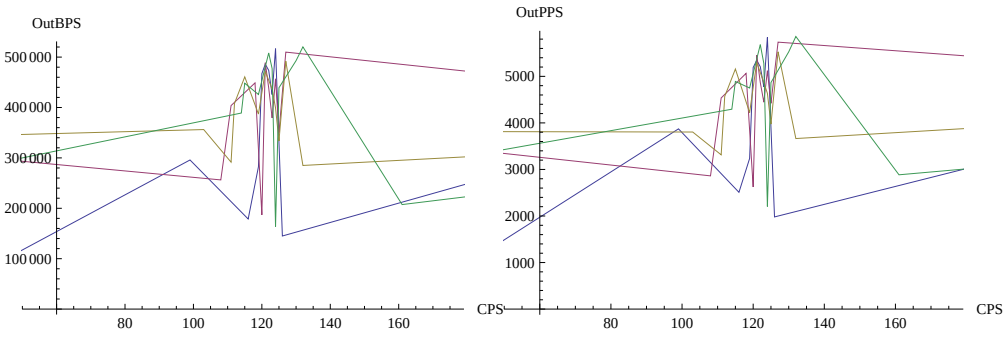
outPPS increases linearly with increase in connections per second. This can be observed from graph (a). Two more real servers are added to the Linux Virtual Server Cluster to see any difference in throughput. Graph (b) shows that with 4 real servers OutPPS remains increases with increase in connections per second but not too long. In first test with 2 real servers max OutPPS was 5000 where as in second test with 4 real serves max OutPPS is about 6000, after that graph shows a straight line with increase in CPS. This indicates that Linux Virtual Server max throughput limit is reached. Then Linux Virtual Server is implemented with 6 and 8 real servers to see any difference. The results of throughput with 6 and 8 real servers are shown in graphs below.



(a) OutPPS6 (b) OutPPS8

Figure 3.10: OutPPS with 6 and 8 real servers

In these tests we observed that Linux Virtual Server throughput did not increase when we jumped from 4 to 6 real servers and then from 6 to 8, the max throughput remain same as with 4 real servers. It can be observed from the graph, that with 6 and 8 real servers, after reaching max throughput limit, it started decreasing with increase in CPS and we can see that it decreases more with 8 real servers then 6 real servers.



(a) OutBPS (b) OutPPS

Figure 3.11: Average OutBPS and OutPPS with 2-8 real servers

Figure 3.11 graph (a) shows combined OutBPS to CPS average result of four tests with varying number of real servers i.e. from 2 to 8 by increasing 2 servers in each test where as graph (b) shows combined OutPPS to CPS average result of four test using

varying number of real servers from 2 to 8, increasing 2 servers in each test.

We got an increase in OutPPS and OutBPS with an increase in total number of connections while using 2 and 4 real servers. But with 4 real servers we saw that after an initial increase in throughput we got straight line showing a constant throughput with increase in CPS. You will see in all graphs that around 100 to 130 CPS, throughput result of all tests gives strange up and down values. I observed that as I have one 10M out of ten files on the servers for downloading from the client to generate heavy traffic on the Director, I observed that when number of connections reaches 100 to 130, the number of requests for downloading 10M file reached to such a number which puts heavy load on the Servers to process and heavy load on the Director to transfer traffic back to clients. This causes director for sudden decrease in throughput. Analyzing all tests results shows that in this scenario 4 real server Linux Virtual Server is the best choice.

3.3.2 LVS via NAT Using Weighted Round Robin Scheduling

The experimental setup of LVS via NAT using Weighted Round Robin scheduling algorithm as the same as shown in figure of LVS with Round Robin scheduling algorithm. Using different data files number of tests will be conducted using Weighted Round Robin scheduling algorithm. In each test we will keep the same basic configuration but we will change the data files used to be downloaded to collect throughput data of light, medium and heavy traffic through the LVS director. In each test we will conduct four different tests by varying the number of real servers used by the Linux Virtual Server.

The common configuration of the Linux Virtual Server in all test will be as below.
Packet forwarding method: -m -masquerading
Scheduling Algorithm: wrr - weighted round robin
The number of real servers will be used from 2 to 8, four sub tests will be made by just adding 2 real servers in each sub test where as data files placed on the real servers for downloading will also changed to generate different set of traffic through the LVS-Director.

Weighed Round Robin is basically designed to perform better in a situation where Linux Virtual Server cluster consists of different capacity real servers. In Weighted Round Robin the higher capacity real servers are assigned higher weight where as lower capacity servers are assigned lower weight. Weighted Round Robin scheduling algorithm then assigns more requests to real servers with higher weight and less request to the servers with lower weight. This weight is basically an integer value which shows capacity of a server[21].

Weighted Round-Robin Scheduling assigns incoming request in sequence to the group of real servers however assigns more tasks to servers which have higher capability. If there is considerable dissimilarity in the capability of real servers group, then Weighted Round-robin scheduling is the best solution. But in our test setup we

have all same capacity real servers. So its not possible to determine efficiency of Weighted Round Robin scheduling algorithm but I am still curious to perform tests using Weighted Round Robin scheduling to know its performance with same capacity real servers. I am expecting that Weighted Round Robin will not give better throughput then Round Robin, as Round Robin is the best scheduling algorithm if Linux Virtual Server contains equal capacity real servers.

Test Case A Method

In Test Case A we will use the basic common configuration as mentioned above. We will conduct four different tests with varying number of real servers from 2 to 8. In first test 2 real servers will be used then 2 real servers will be added to perform the next test. Number of tests with varying number of real servers gives used the throughput performance of Linux Virtual Server with different set of real servers.

SetupLVS.sh script given in Appendix is used to configure Linux Director for a given number of real servers to forward incoming requests from clients directed towards port 80 on 128.39.73.11, to port 80 on given number of real servers in the server pool. NAT or masquerading is used as packet forwarding method and Weighted Round Robin is used as scheduling algorithm. In this case we are using NAT or masquerading as forwarding method, so we need to set default gateway of all the real servers to the internal interface of the Director, which is connected to the LAN. The following command will set Directors internal interface as default route of the real servers.
echo ”1” $>$ /proc/sys/net/ipv4/ip_forward
Client.pl script given in Appendix is used to download different web pages continuously in a random manner. Four web pages of size 76K, 103K, 505K and 1.8M are placed on all real servers. Client.pl script is used to run on four different clients to download these pages randomly to generate different size of traffic on the Load Balancer. Crontab is used to save ipvsadm throughput-stats, rate-stats and connection-stats. Crontab is set to save ipvsadm stats data after every minute to three different files. These files are used to analyse data and then to generate throughput graphs afterward

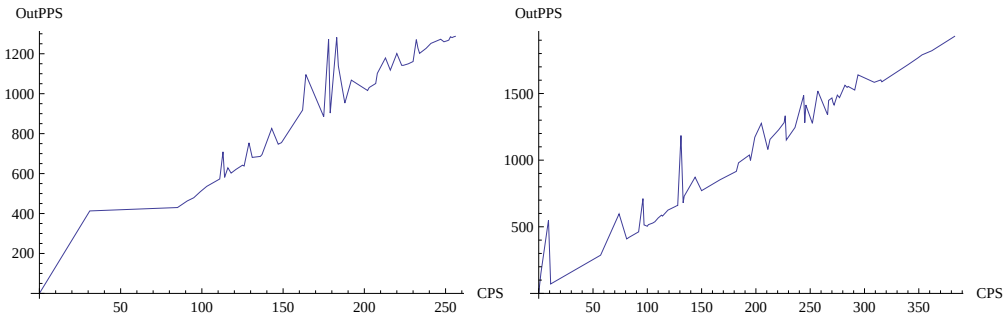
Test Case A Results

For Test Case A first Linux Virtual Server is configured with two real servers but after getting results, two more real servers were added to the cluster and then third and fourth time two more servers were added. In this way we got throughput data of four different sub tests with varying number of real servers from 2 to 8. Crontab is used to save different stats data on director. . These files are mentioned below.

Rate-stats-2 to rate-stats-8. These files are used to store Input packets per second, output packets per second, input byte per second, output byte per second and connections per second to director. These files store stats data of Director as well as of the real servers used. Setup is used to run for 2 to 8 real servers and crontab used to store rate stats respectively in four different files. Similarly two more files are used for storing other stats information, Throughput-stats-2 to throughput-stats-8 are used to save total number of connections, total input packets, total output packets, total input bytes and total output bytes where as Conection-stats-2 to conection-stats-8 are used to store total number of connections with clients IP address, Director IP address and IP address

of the chosen real server for that client's request.

Data in these files are organized in such a way to import in Mathematica for analysis and generating graphs These graphs are given as below.

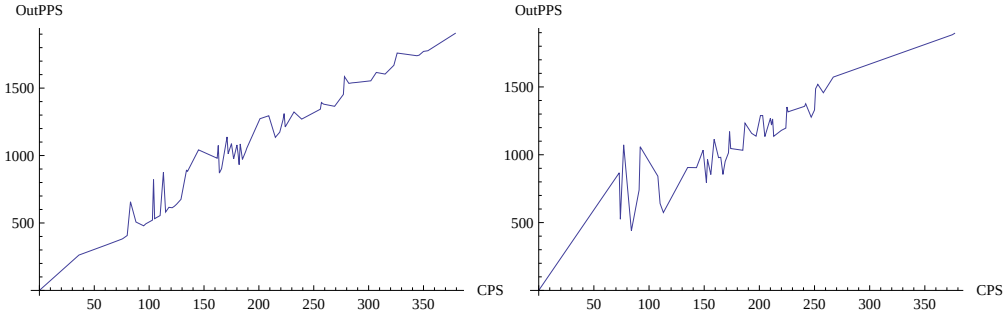


(a) OutPPS2 (b) OutPPS4

Figure 3.12: OutPPS with 2 and 4 real servers

Graph in Figure 3.12 shows two graphs one with 2 and one with 4 real servers. Both graphs show CPS to outPPS. Graph (a) shows that outPPS continuously increases with increasing total number of connections per second. Then 2 more real servers are added to the Linux Virtual Server cluster and graph (b) shows results of LVS with 4 real servers. You can see that output packet rate per second increases initially with an increase in CPS.

Keeping in mind that in this test we have placed the same files on the real servers for download .which were placed in Test Case A with Round Robin scheduling algorithm. So we will compare results of this test with the result of Test Case A of the Round Robin scheduling. It has been seen that adding 2 more real servers gave an increase in output PPS from about 1200 PPS to 1600 PPS but after that throughput is not increasing. To confirm this two more test are conducted with 6 and 8 real servers and has been seen that there is no difference of throughput from 4 to 6 and then from 6 to 8 real servers. Results of these tests are given below.



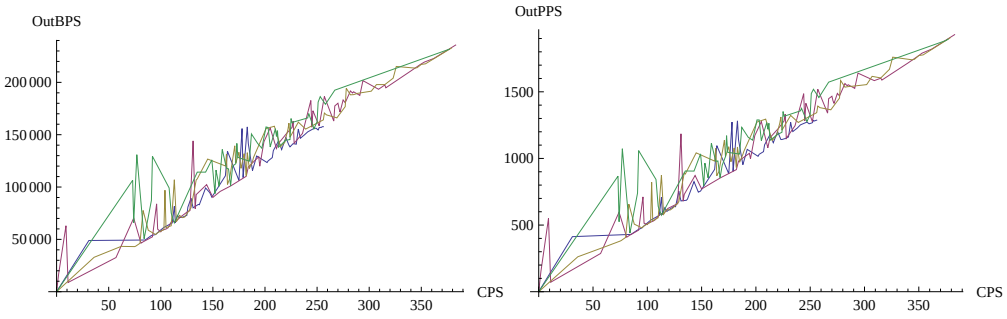
(a) OutPPS6 (b) OutPPS8

Figure 3.13: OutPPS with 6 and 8 real servers

Graph (a) shows result of LVS outPPS with 6 servers where as graph (b) shows results of LVS outPPS with 8 real servers. You can see from the results that there

is no difference of throughput with 6 and 8 real servers. So these results show that using weighted Round Robin scheduling and placing small data files on real servers for downloading initially gave us an increase in outPPS throughput with 2 and 4 real servers and then adding more real severs to the cluster doesn't cause an increase in throughput.

If we compare this test result with the same test with Round Robin scheduling, we see that using Round robin scheduling having placed small files on the real servers for download, LVS did not reached the Director's bottleneck and even with 8 real servers we had an increase in throughput with increase in total number of connections per second where as using Weighted Round Robin scheduling algorithm LVS Director reached to its maximum throughput support with a link of 100M. These results proved that Round Robin is better the Weighted Round Robin for downloading small data files and using equal capacity real servers in the Linux Virtual Servers cluster.



(a) OutBPS (b) OutPPS

Figure 3.14: Average OutBPS and outPPS with 2 - 8 real servers

Below Figure 3.14 shows an average graph of all four tests i.e. with 2, 4, 6 and 8 real servers. Graph (a) shows outBPS and graph (b) shows outPPS with respect to CPS for four tests together.

All graphs shows almost same result. It has been seen that with Weighted Round Robin all throughput rate were not increasing continuously but there have been up and down in throughput rate, where as with Round Robin scheduling we had continues increase in throughput and not much up and down throughput rate.

So in this scenario Weighted Round Robin is not the best choice, it would be nice to test weighted Round Robin scheduling having different capacity real servers in the Linux Virtual Server cluster to check if weighted Round Robin performs better in that scenario.

Test Case B Method

Test Case B method is same as Test Case C of the Round Robin scheduling. The Only difference is the scheduling algorithm used. Test Case C is designed to place smaller and medium data files on all real servers. These data files are from 100K to 10M sized.

These data files can be taken as small website having some small text and pictures files and one or two audio files of up to 10M.

setupLVS.sh script given in appendix is used to setup Linux Virtual Server. Test C will be conducted four times for 2, 4, 6 and 8 real server's setup to check throughput of Linux Virtual Server having placed the above given data files for download from the client's end. The same scheduling algorithm, Weighted Round Robin in this test as with Test Case A.

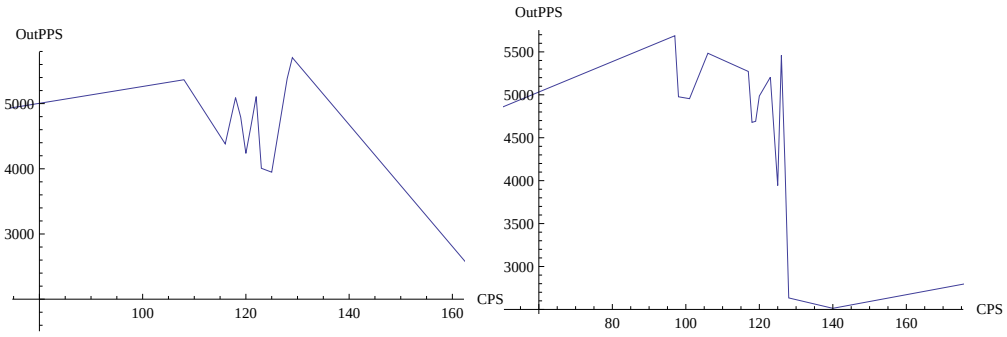
Test Case B Results

Client.pl script is used to run on four different clients. This script randomly request downloading the mentioned data files from the servers. All client sends request to the Director and Director chooses one of the real server from the cluster for handling this request. Director chooses real Server on the basis of scheduling algorithm.

As in this test we are using Weighted Round Robin scheduling algorithm. This scheduling algorithm assigns more requests to the servers with higher weight and fewer requests to the servers with lower weight. But in our case we are using same real servers for handling client's request. So these servers have equal weight. In test Case A we checked that with smaller data files this scheduling algorithm did not work well as compared to Round robin. Now we will check throughput of Linux Virtual Server with these mixed files and compare result with Round Robin to determine if it works better in this scenario.

Crontab is used to collect throughput stats data in different files. Crontab is set to save throughput stats data after every minute. These data files are reorganised in such a way to import in Mathematica for analysis and generating graphs of inn and out byte per second as well as inn and out packets per second.

Using Mathematica input and output rate per second and input and output packets per second to total number of connections per second graphs are generated. Using average function of Mathematica, all average graphs were also generated. Some of these graphs are given below



(a) OutPPS2 (b) OutPPS4

Figure 3.15: OutPPS with 2 and 4 real servers

Figure 3.15 shows two graphs. Graph (a) shows outPPS to CPS with 2 real servers where as graph (b) shows same statistics with 4 real servers. When we compare these results with Test Case C of the Round Robin, we see that weighted Round Robin gives bad throughput results as compared to Round Robin because in Round Robin from 2 to 4 real servers throughput was increasing with increase in CPS where as in Weighted

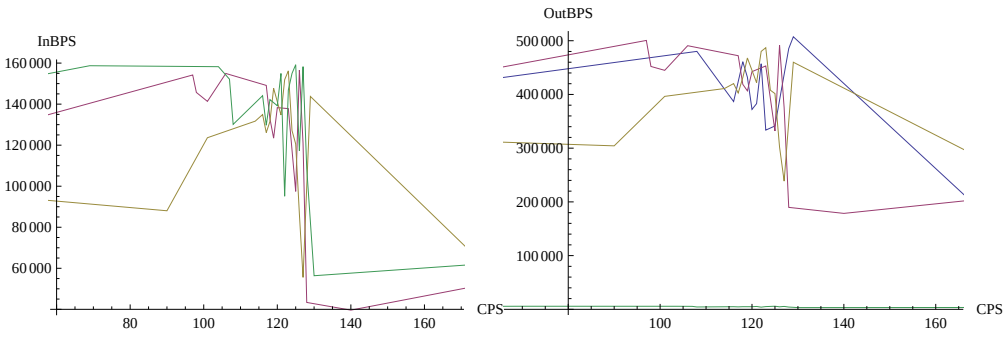
Round Robin we can see that even with 2 real servers initially outPPS starts around 5000 but later with increase in CPS this number decreases continuously. This decrease in throughput remains continues even when we setup 4 real servers to the Linux virtual Server cluster and same result with 6 and 8 real servers.

We observed from these test that throughput did not increased when we added more real servers to the Linux Virtual Server. As we have equal capacity real servers, it means that we have equal weighted real servers. So in his case Weighted Round Robin should work same as Round Robin. But results show that Weighted Round Robin throughput performance is worse then Round Robin with equal capacity real servers.

Figure 3.16 graph (a) shows combined InBPS to CPS average results of four tests with varying number of real servers i.e. from 2 to 8 real server setup through increasing 2 real servers in each test where as graph (b) shows combined OutBPS to CPS average results of four test.

We observed all the graphs and concluded that only initially all throughput stats starts from a higher value but then with increase in connections per second, all throughput stats rate continued to decrease. You will see in all graphs that around 130 to 140 CPS, all throughput rate produces strange up and down results. This is same as we observed with Round Robin scheduling. As we have placed one big file of about 10M on the real servers. It looks that total number of connections reaches about 130 to 140 per second, and then the random number of requests for downloading this big file increases to such a number, that all servers takes time to process this big file. As a result output throughput and input throughput decreases dramatically.

From Test Case A and Test Case B we concluded that having equal capacity real servers in the Linux Virtual Server cluster, Weighted Round Robin scheduling is not the best choice for maximum throughput and Round Robin scheduling method produces better throughput results then Weighted Round Robin scheduling.



(a) InBPS (b) OutBPS

Figure 3.16: Average InBPS and OutBPS with 2 to 8 real servers

3.3.3 LVS via NAT Using Least Connection Scheduling

Least Connection scheduling algorithm works keeping track of all established connections to real servers. It uses IP Virtual Server table for storing established connections

of the real servers. In this way least Connection scheduling algorithm sends more requests to servers which have less active connections. Least Connections performs better if network traffic have varying load and if the real servers in the Linux Virtual Server pool are of same capacity. This makes Least Connections a suitable choice when Linux Virtual Server Cluster has almost same capacity real servers and a mixed type of traffic load.

Initially it can be seen that least connection algorithm also performs well with real servers having different capacity. Actually initially higher capacity servers receive more requests. But With the passage of time performance will decrease.

Experimental setup of LVS Server through Least Connection scheduling algorithm is same as shown in Round Robin and Weighted Round Robin. The only difference will be the scheduling algorithm. Different sized data files will be used for downloading from the clients to generating light, medium and heavy traffic through the Linux Virtual Server Director. Using different data files multiple experiments will be conducted keeping the same configuration. Different experiments will be taken using varying traffic load. First very light traffic will be used and then a mixed load traffic will be used to check either Least Connection performs better, as we have same capacity real servers so least connection should perform better according to theory.

By varying the number of real servers four different tests will be conducted. The basic configuration will be as given below.
Packet forwarding method: -m -masquerading
Scheduling Algorithm: wrr - weighted round robin
The number of real servers will be used from 2 to 8, four sub tests will be made by just adding 2 real servers in each sub test where as data files placed on the real servers for downloading will also changed to generate different set of traffic through the LVS-Director.

Test Case A Method

Using Least Connection scheduling we will conduct same experiments as with Round Robin and weighted Round Robin. In Test Case A we will use the common configuration of the Linux Virtual Server as mentioned above. We will do experiments of four different tests using varying number of real servers i.e. from 2 to 8. First we will use 2 real servers and then we will continue just adding 2 more real servers for the second test and so on. Different tests with varying number of real servers give us throughput of Linux Virtual Server with different number of real servers.

setupLVS.sh script given in Appendix is used to setup Linux Virtual Server for given number of real servers and with given scheduling algorithm. Here we are using Least Connection scheduling algorithm. Director is configured to forward incoming requests from clients directed towards port 80 on Director to port 80 on the chosen real server in the Linux Virtual Server pool. Packet forwarding is done via NAT or masquerading. And for masquerading we need to set default gateway of the real servers to the internal interface of the Director.

Client.pl script given in Appendix used to run on all four clients for downloading the files placed on the real servers. Client.pl send request randomly for downloading these files. In Test Case A we are using very light files. The files size will be same as used in Round Robin and Weighted Round Robin Test Case A i.e. files of size

76K,103K,505K and 1.8M .

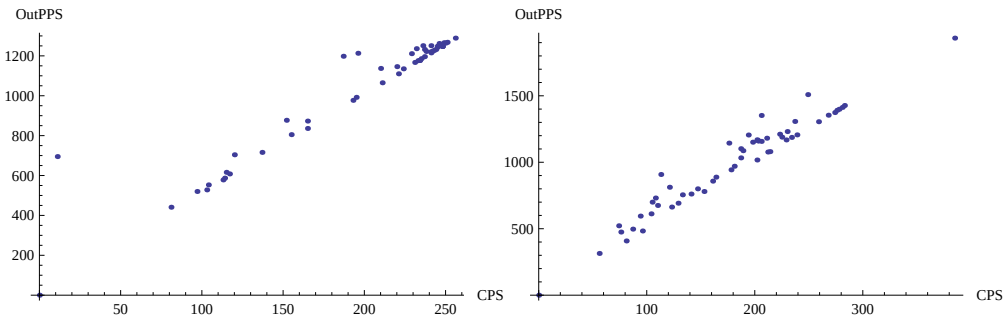
Crontab is to used save ipvsadm throughput stats, rate stats and connection stats for all tests in different files. Crontab saves ipvsadm stats information after every minute to three different files and for each test. Data is organized in these files to make possible for mathematica to iport data from these files for generating different throughput graphs.

Test Case A Results

Client.pl script is used to run from four diferent clients for downloading data files from the Linux Virtual Server. This generates traffik on the Linux Virtual Server Director. Ipvsadm is used to check throughput of the director. Crontab will run ipvadm to get throughput after every minute and then stores this throughput into three different files. For all tests from real servers 2 to 8 crontab is set to store throughput stats respectively.

All useful stats data is collected and formatted for the mathematica to import these files and generate different throughput stats graphs. Some of these graphs are given below for reference.

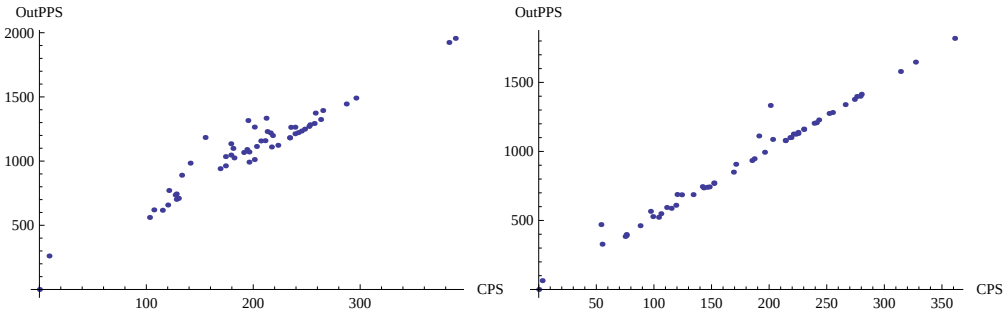
Figure 3.17 shows two graphs. Graph (a) shows outPPS using 2 real servers web services Linux Virtual Server where as graph (b) shows same result of 4 real servers. outPPS to CPS grows linearly. With increase in CPS, outPPS throughput increases continuously. This increase in outPPS throughput continues increasing even with 4 real servers web services Linux Virtual Server. A significant amount of increase in outPPS to CPS can be seen in graph (b), which shows that LVS throughput increased with increasing number of real servers.



(a) outPPS2 (b) OutPPS4

Figure 3.17: Output PPS with 2 to 4 real servers

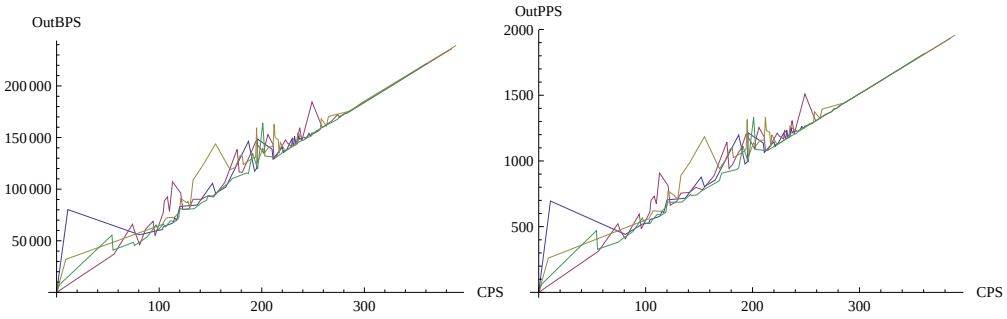
I got an increase in throughput by adding more real server, then two more tests with 6 and 8 real servers are implemented and throughput analysis shows that all throughput increased by adding 2 more real servers. But it has been seen that with 8 real servers there is not much difference in throughput from 6 real server's setup. As we are using small sized files for downloading from four clients' continuously in a loop. We will compare this result with Round Robin Test Case A and Weighted Round Robin Test Case A. when we compare this result with previous scheduling method results, then we see that, Least Connection scheduling algorithm gives higher throughput then Weighted Round Robin and It works almost same as Round Robin scheduling. I was



(a) outPPS6 (b) OutPPS8

Figure 3.18: Output PPS with 6 to 8 real servers

expecting Least Connection will give higher throughput then Weighted Round Robin, as Least Connection algorithm performs better with equal capacity real servers. We are using same capacity real servers and you can see through results of this test that Least Connection gives higher throughput then previous weighted Round Robin and works almost same as Round Robin. When we switch from 6 to 8 real servers, we see initially an increase in throughput with increasing total number of connections per second, but afterwards throughput not increasing more then 2000 outPPS. This shows that Director's bottleneck is reached and no more throughputs can be increased by just adding more real servers. Figure 3.18 shows results of outPPS to CPS with 6 and 8 real servers.



(a) outBPS (b) OutPPS

Figure 3.19: Average OutBPS and OutPPS with 2 to 8 real servers

Figure 3.19 shows average throughput graphs of all four tests, using 2, 4, 6 and 8 real servers. Graph (a) shows outBPS and graph (b) shows outPPS with respect total number of connections per second for all four tests together.

The above graphs shows an average of each graph. Mathematica is used for generating average throughput stats graphs. Average graph is generatd for all 2, 4, 6 and 8 real servers setup. All average graphs shows that when we increase number of real servers, number of connections per second increases and as well as outPPS, outBPS, inPPS and inBPS increases.which shows that Linux Virtual Server performs very well with the given load of the downloaded files til 6 real servers. As we increase real

servers Director disributes request to more servers, which increases throughput as a result.

You will not see any difference in average graphs of Round Robin and least Connection scheduling algorithms. Both of these algorithm works almost same for HTTP services. It could be interesting to see with other services but here I am using only HTTP services so in my case both these scheduling algorithms performs almost equally and both performs better then Weighted Round Robin.

Test Case B Method

In test case B of Least Connection scheduling we will use the same method as used in Test Case B of Weighted Round Robin and Test Case C of Round Robin scheduling algorithm. The only difference will be the scheduling algorithm, as here we are using Least Connection scheduling. Test Case C is basically designed to place mixed sized data files on real servers for downloading. These data files are from 100K to 10M data sized

SetupLVS.sh script given in appendix is used for setting Linux Virtual Server for given number of real servers and given scheduling algorithm. Test C will be run for four times for 2, 4, 6 and 8 real servers. Different numbers of real servers are used to check throughput performance with increasing real servers. We are using equal capacity real servers and a mixed sized traffic will be generated. We will see how Least Connection scheduling performs in this condition.

Test Case B Results

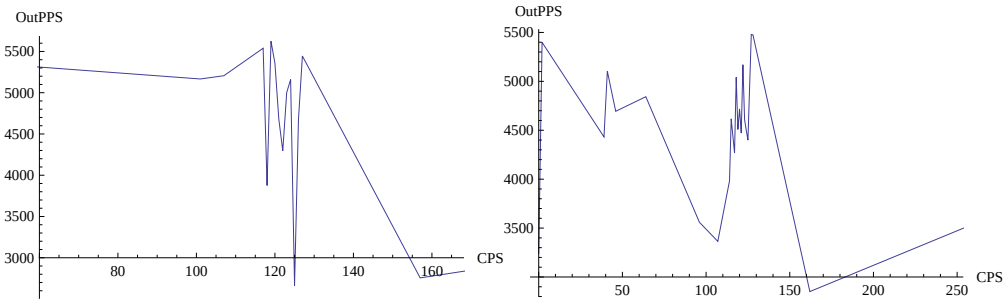
Linux Virtual Server is configured with scheduling algorithm Least Connection and given number of real servers with setupLVS.sh script. This test is also used to run four times for collecting all throughput stats data of Linux Virtual Server from 2 to 8 real servers.

Crontab is used to save all throughput stats results from all four tests to four different files. These results are organised in different files to import in Mathematica for generating different throughput stats graphs.

Mathematica generates different Rate stats to CPS graphs and also averaged results graphs are generated to compare this test result with previously used Round Robin and Weighted Round Robin scheduling. Some of these graphs are given below.

Graph (a) shows outPPS to CPS configuring 2 real servers LVS where as graph (b) shows outPPS to CPS configuring 4 real servers LVS. You can see that outPPS throughput stats remained almost same in 2 and 4 real server's setup. When we compare this test result with Round Robin scheduling, we see that in Round Robin there was a slight increase in outPPS rate when we jumped from 2 to 4 real server's setup. But in Least Connection we see almost same throughput result with 2 and 4 real servers, but we can see that maximum throughput of Round Robin with 4 real servers and least connection with 2 real servers is also same.

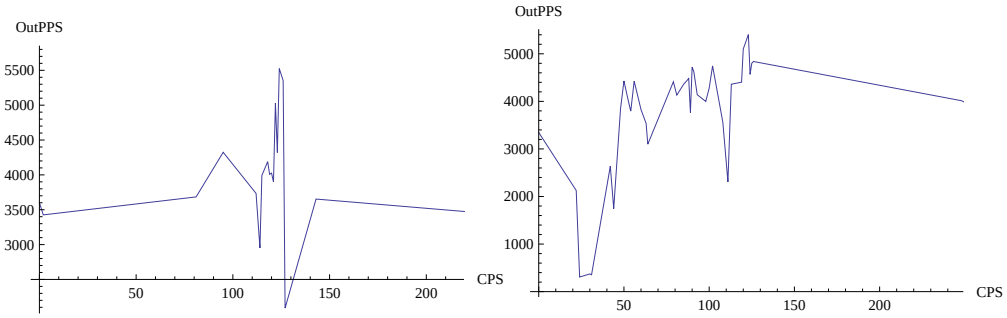
This shows that using Least Connection scheduling LVS already reached the maximum throughput limit provided 100M link to the Director using 2 real servers. Two more tests are implemented using 6 and 8 real servers to see if there any difference in



(a) outPPS2 (b) OutPPS4

Figure 3.20: Output PPS with 2 and 4 real servers

throughput by increasing real servers. Result of throughput with 6 and 8 real servers is shown in graphs below.



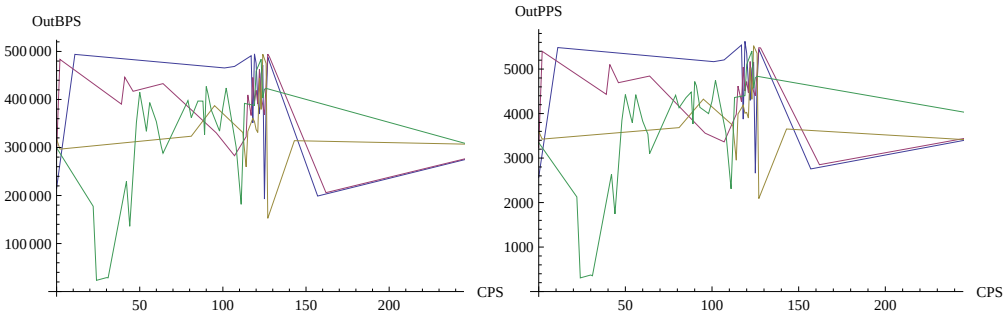
(a) outPPS6 (b) OutPPS8

Figure 3.21: Output PPS with 6 and 8 real servers

We can see that throughput rate did not increased when we implemented LVS with 6 and 8 real servers and the maximum outPPS rate remains same as with 2 and 4 real servers. Thus this result shows that with given traffic load, 100M link to the director and given real servers, LVS with 2 severs is enough to reach maximum performance in this scenario. There is no difference in throughput from 4 to 8 real servers, so there is no reason for using more then 4 real servers LVS.

Figure 3.22 graph(a), represents combined OutPS to CPS average results of all four tests with different number of real servers, i.e. from 2 to 8 real servers by adding 2 more real servers for the next test. Graph (b) represents combined OutPPS to CPS averaged result of all four tests.

We observed that there is increase in OutPPS and OutBPS with the increase in total number of connections while using 2 real servers in the Linux Virtual Server cluster. With 4 real servers there a slight increase in out rates, but not significant and we can say that overall throughput with 4 real servers is same as with2 real servers. Similarly with 6 and 8 real servers there is no improvement of output rate stats, so we can say that even with 2 real servers LVS Director reached its bottleneck and throughput can't be increase by just adding more real servers to LVS cluster.



(a) avrOutBPS (b) avrOutPPS

Figure 3.22: Average OutBPS and OutPPS with 2 to 8 real servers

You will see another similarity in results with Round Robin and weighted Round Robin is that between 100 to 130 CPS, all graphs gives very up and down results. It bas been concluded that when number of connections per second reached about 100 to 130, the number of requests for 10M size files increases certain number, that creates heavy load on the real servers for processing as well as a heavy load traffic on the Director to transfer packets back to the client.

This creates up and down in throughput. Analysing all tests graphs, it can be concluded that 2 or 4 real servers LVS with Least Connection scheduling is best choice in this scenario. It can also be concluded that Least connection scheduling defiantly gives better performance then weighted Round Robin and almost same performance as compared to Round Robin by using equal capacity real servers in the Linux Virtual Server cluster.

Chapter 4

Conclusion

We have discussed how to create LVS (Linux Virtual Servers) using a number of real servers and a frontend Director. We have explored three different Load Balancing techniques NAT (Network Address Translation), direct routing and IP Tunneling, which can be used for making parallel services of the Linux Virtual Server cluster to appear for the end user just as a virtual service on an IP address. We have discussed four different scheduling algorithms which LVS supports.

In our experimental setup, we have used NAT (Network Address Translation) as Load Balancing Technique and we have done experiments with three different scheduling methods Round Robin, Weighted Round Robin and Least Connection. We have also generated three different types Load traffic on the Linux Virtual Server Director. Also varying number of real servers has been used in the experiments to check Linux Virtual Server performance with different number of real servers. Here we will conclude the performance of each scheduling method one by one.

- LVS-via NAT with Round Robin Scheduling

Our three experiments using Round Robin scheduling method shows that, with light Load traffic Round Robin scheduling works fine and we get increase of Input rate and output rate stats with increase in connection per second. We checked this throughput with varying number of real servers and results show that with the very light load traffic, Linux Virtual Server Director can easily manage to accept new connections and distributing them towards real servers in the server pool.

Experiments with heavy load traffic shows that Linux Virtual Server throughput increases initially with an experiment with only two real servers but experiments with 4, 6 and 8 real servers shows that there is no increase in throughput rate and connections per second by just adding more real servers to the Linux Virtual Server cluster. This is because LVS Director reached its bottleneck of 100M link and it reached the maximum possible throughput with 100M link.

Experiments with mixed load traffic shows that Linux Virtual Server throughput increases with 2 and 4 real servers' setup but after that with 6 and 8 real servers there is no increase in throughput. The only difference with mixed load traffic is that Director reached its maximum possible throughput with 4 real servers where as with heavy traffic load it reached with 2 real servers and the over maximum throughput in same in both experiments. The reason of increased in throughput with 4 real server is that we

have mixed load traffic, light, medium and heavy where as in previous experiments we have used heavy load traffic.

- LVS-via NAT with weighted Round Robin Scheduling

We have done experiments with weighted Round Robin using light and mixed load traffic. With light load traffic Linux Virtual Server using Weighted Round Robin results show that throughput increases with 2 and 4 real servers but after that with 6 and 8 real servers' throughput didn't increase and we got constant throughput with increase in real servers. Comparing these results with Round Robin scheduling shows that Round Robin scheduling is much better then weighted Round Robin in our experimental setup.

With mixed load traffic the results are even worse and even with two real servers initially throughput starts from higher value but then continually decreases with the increase in CPS. This has test result is even worse with 4, 6 and 8 real servers. Instead of increase in throughput we got decrease with adding more real servers to the Linux Virtual Server cluster. This result was expected as Weighted Round Robin is suitable only if there are different capacity real servers but in our experiments we are using all same capacity real servers, so defiantly Round Robin performs better then Weighted Round Robin scheduling.

- LVS-via NAT with Least Connection Scheduling

With Least Connection scheduling we have also done experiments with light and mixed load traffic. With light load traffic Linux Virtual Server using Least Connection scheduling results are exactly same with the results using Round Robin scheduling algorithm. Throughput rate increases continuously increases with the increase in CPS and this increase in throughput continues with 4, 6 and 8 real servers. This result shows that with light traffic load using Least Connection scheduling Linux Virtual Server Director manages to accept new connections and servers can easily manage to handle these requests as a results over all throughput increases continuously with increase in connections per second.

Mixed load traffic results show that, Round Robin performs little bit better then least Connection but over all throughputs remains same with Round Robin and Least Connection scheduling.

Analyzing all results with three scheduling algorithms, I concluded that in our experimental setup and with given 100M link to Director as well as with given network traffic load, Round Robin and Least Connection scheduling with NAT (Network Address Translation) performs equally better then Weighted Round Robin.

We did not experience any significant difference in throughput using Round Robin and Least Connection. There might be differences with other services but we are only using HTTP services. Weighted Round Robin must perform better with a varying capacity of real servers of a Linux Virtual Server cluster. But we couldn't confirm it through experiment, as we have all same capacity real servers. But our experiments using Weighted Round Robin scheduling proved that it's not best choice to use Weighted Round Robin having equal quality real servers.

Appendixes

LVS SetUp Script

This is a bash script used to setup Linux Virtual Server. It prompts user to chose a scheduling algorithm and number of real servers. First of all it clrears all stats counters so that to start counting from zero and then it sets the given scheduling algoritham for the Director and also adds the given number of real servers to the Linux Virtual Server Cluster. The routing method used is NAT. As NAT uses IP Forwarding so it also enables IP Forwading on the Director. The maximum number of real servers that can be used is 8. The script code is given below.

```
##This script will setup LVS with given nr of real servers##
##and scheduling Algorithm##
#############################################################
 clear;
 echo -e "Enter scheduling method: \c "
 read  m
 echo "The scheduling method entered: $m"
 echo -e "Enter number of real servers(2,4,6 or 8): \c "
 read  s
 echo "Number of real servers  entered: $s"
 case $s in
 2)
  echo "setting up LVS-$s with 2 real server....."
  /sbin/ipvsadm -C
  /sbin/ipvsadm -A -t 128.39.73.11:80 -s $m
  /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.254:80 -m
  /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.253:80 -m
  echo "1" >/proc/sys/net/ipv4/ip_forward
  /sbin/ipvsadm --zero
  ;;
 4)
  echo "setting up LVS-$s with 4 real server....."
  /sbin/ipvsadm -C
  /sbin/ipvsadm -A -t 128.39.73.11:80 -s $m
  /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.254:80 -m
  /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.253:80 -m
  /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.252:80 -m
```

```
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.251:80 -m
 echo "1" >/proc/sys/net/ipv4/ip_forward
 /sbin/ipvsadm --zero
 ;;
 6)
 echo "setting up LVS-$s with 6 real server....."
 /sbin/ipvsadm -C
 /sbin/ipvsadm -A -t 128.39.73.11:80 -s $m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.254:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.253:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.252:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.251:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.250:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.249:80 -m
 echo "1" >/proc/sys/net/ipv4/ip_forward
 /sbin/ipvsadm --zero
 ;;
 8)
 echo "setting up LVS-$s with 8 real server....."
 /sbin/ipvsadm -C
 /sbin/ipvsadm -A -t 128.39.73.11:80 -s $m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.254:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.253:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.252:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.251:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.250:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.249:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.248:80 -m
 /sbin/ipvsadm -a -t 128.39.73.11:80 -r 10.255.255.247:80 -m
 echo "1" >/proc/sys/net/ipv4/ip_forward
 /sbin/ipvsadm --zero
  ;;
 *)
 echo "setupLVS.sh "
 ;;
esac
```

LVS Client Script

Client script is used to run on four differnt clients for randomly downloading files placed on the real servers. These files have been changed for light, heavy and mixed load traffic. The script runs in a loop and for a given number of max count. wget command is used to download a random file from the Linux Virtual Server.Script code is given below.

```
@files=("file0","file1","file2","file3","file4",
```

```
        "file5","file6","file7","file8","file9");
$maxcount=200000;
$count = 0;
while($count<$maxcount)
{
  $number=int(rand(10));
  $file="$files[$number]";
   system("/usr/bin/wget -O testfile -q http://128.39.73.11/data/$file");
   unlink testfile;
    $count++;
     if($count % 100 == 0)
      {
        print "Number of request sent : $count \n";
         # $count = 0;
      }
     #sleep 1;
  }
```

LVS Setup Crontab

cron is a unix, utility which makes possible for the tasks to run automatically with a given interval of time by the cron daemon. This file contains crontab settings. crontab can be used to store data with a given regular intervals of time. With * setting, crontab stores after every minute, the result of each command to a given file. In all experiments crontab is set to store throughput stats, rate stats and connection stats in three different files. These files are used to extract stats information for data analysis and generating graphs. The Crontab setting is given below.

```
##This file contains crontab setting.##
* * * * * echo"LVS Stats"
* * * * * /sbin/ipvsadm -L -n --stats >>/home/s137507
          /results/lvs-rr/throughput-stats--2
* * * * * /sbin/ipvsadm -L -n --rate  >>/home/s137507
          /results/lvs-rr/rate-stats--2
* * * * * /sbin/ipvsadm -L -n --connection  >>/home/s137507
          /results/lvs-rr/conection-stats--2
```

Bibliography

[1] LVS Introduction-Load Balancing Server Cluster
http://www.linuxvirtualserver.org/whatis.html

[2] Patrick O'Rourke and Mike Keefe, *"Performance Evaluation of Linux Virtual Server"*, Mission Critical Linux, Inc.

[3] Wensong Zhang, *"Linux Virtual Server for Scalable Network Services"*, National Laboratory for Parallel & Distributed Processing Changsha, Hunan 410073, China.

[4] Horms (Simon Horman), *" Linux Virtual Server Tutorial"*
http://www.ultramonkey.org/papers/lvs_tutorial/html

[5] Eric Searcy and Tag1 Consulting, *"Scalable Linux Clusters with LVS, Considerations and Implementation, Part I"*

[6] Linux Virtual Server Administration, *"Linux Virtual Server (LVS) for Red Hat Enterprise Linux"*
http://www.centos.org/docs/5/html/5.2/Virtual_Server_Administration/index.html

[7] Amir Ranjbar, Keith Hutton, *"Designing High-Availability Services"*,CCDP Self-Study

[8] Red Hat Cluster Suite: Configuring and Managing a Cluster
http://www.redhat.com/docs/manuals/csgfs/browse/rh-cs-en/s1-lvs-scheduling.html

[9] Kineo Open Source, *"Open Source Software testing tools and discussion"*
http://www.opensourcetesting.org/performance.php

[10] Apache Software Foundation, *"Apache Jakarta Project"*
http://jakarta.apache.org/jmeter/

[11] OpenWebLoad, *"Web Application testing tool"*
http://openwebload.sourceforge.net/

[12] Pylot: web Performance, *"Web Services Performance & Scalability Testing tool"*.
http://www.pylot.org/

[13] Rohit Girhotra on June 09, 2005, *"Building a Linux virtual server"*
http://www.linux.com/feature/45236

[14] iptables(8) - Linux man page, *"administration tool for IPv4 packet filtering and NAT"*
http://linux.die.net/man/8/iptables

[15] ipvsadm(8) - Linux man page, *"Linux Virtual Server administration"*
http://linux.die.net/man/8/ipvsadm

[16] David A. Ranch, November 13, 2005 *"Linux IP Masquerade HOWTO"*
http://tldp.org/HOWTO/IP-Masquerade-HOWTO/index.html

[17] Mohit Aron - Darren Sanders - Peter Druschel - Willy Zwaenepoel, *"Scalable Content-Aware Request Distribution in Cluster-Based Network Services"*
Department of Computer Science, Rice University, Houston, TX 77005
http://db.usenix.org/events/usenix2000/general/aron/aron_html/

[18] LVS Introduction, *"Load Balancing Server Cluster"*
http://www.iterasi.net/openviewer.aspx?sqrlitid=bbetrztneuof7s6mljz83a

[19] Joseph Mack, *"LVS-HOWTO"*
http://www.ntua.gr/lvsp/Joseph.Mack/HOWTO/

[20] http://kb.linuxvirtualserver.org/wiki/LVS/DR

[21] http://kb.linuxvirtualserver.org/wiki/Weighted_Round-Robin_Scheduling

[22] http://gd.tuwien.ac.at/infosys/servers/w3lvs/scheduling.html