Performance Assessment of High Availability Clustered Computing using LVS-NAT

*Muhammad Kashif Shaikh, **Muzammil Ahmad Khan and ***Mumtaz-ul-Imam

***Abstract* — High availability cluster computing environment attempts to provide high availability to computing services. This paper evaluates building and investigating a highly available computing environment that provides a solution to achieve high availability. A prototype of cluster computing environment is developed in Linux environment to provide a single but highly available point of entry. The cluster of computers run web based application to provide services to HTTP users.**

Received: December 09, 2009, Accepted: January 27, 2010

***Key Words*—High availability, Cluster computing, Linux Virtual Server (LVS).**

I. INTRODUCTION

The world is witnessing an explosive growth in computing resources, in terms of network resources and heterogeneity of devices. The increasing popularity of handhelds and web-enabled cell-phones is allowing the mobile individual to access data on the Internet, on the corporate network and at home. The rapid development in mobile digital devices is generating more multimedia rich content and inflating data sizes. In addition, the massive growth of the Internet is pushing computing to a truly network-centric paradigm [1].

A. CLUSTERED SYSTEM

The major goal of this research is the implementation and investigation of variety of Linux Virtual Server (LVS) with different scenarios. It also investigates the limitation of director server and real server with different load. In this approach a highly available computing environment using ordinary computer systems for the development of a clustered system is required, which maintains the state of operation even in the event of a failure. This computing environment complies with following features:

- Guaranteed resource availability for critical applications
- Reliability and Transparency

A Server Cluster is basically a group of servers that function as one. By doing so, client machines see the service or application processed by the cluster as though it were coming from a single machine. There are three basic advantages of running a server cluster i.e. scalability, high availability and manageability [2].

B. LINUX CLUSTER

The Highly Available Linux Virtual Server Cluster is based either directly on contributions from the Linux community, or on components that were inspired or enriched by various Linux community projects. The primary source of the LVS cluster is Wensong Zhang’s Linux Virtual Server (LVS) kernel routing algorithm [3]. The capabilities of the LVS project that the LVS cluster currently supports are:

- Routing service requests from virtual servers to a pool of real servers.
- Load-balancing.
- Packet-forwarding
- Persistent connections.

II. FUNCTIONAL DESCRIPTION OF SYSTEM COMPONENT

As the businesses grow and more people tend to use the internet for performing their daily life operations that are important and critical, the administrators need to have a system that offers reliability, cost effectiveness and high availability even in the case of disaster. A single server that was more than enough a decade ago is not sufficient today, it cannot handle the ever increasing aggressive load of users. Even if the server is upgraded it also has a single point of failure that is not sufficient for business critical environments that rely on the reliability of these servers.

A. ARCHITECTURAL MODEL

The Tri-Cluster Architecture involves three major cluster divisions namely, the load-balancing (director) servers, the real servers and the file/storage servers. In a Tri-Cluster Design every cluster has its own fail-over service and redundancy method, which prevents every cluster from being phased out in an event of a catastrophic disaster. As a solution of the above problems, cluster of servers connected via a fast network are being implemented everywhere. This type of architecture offers more scalability, reliability and cost effectiveness as shown in fig. 1 and fig. 2.

* *Department of Computer Engineering, Sir Syed University of Engineering and Technology, Karachi, Pakistan.* mkshaikh@gmail.com
** *Department of Computer Engineering, Sir Syed University of Engineering and Technology, Karachi, Pakistan.* muzammil.ssuet@yahoo.com
*** *Department of Computer Science, Sir Syed University of Engineering and Technology, Karachi, Pakistan.* imam@ssuet.edu.pk

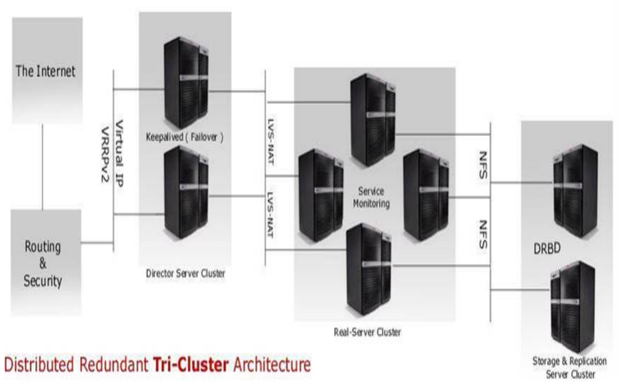

Fig. 1: Distributed Redundant Tri-Cluster Architecture

B. LINUX VIRTUAL SERVER

The Linux Virtual Server is one solution that meets the requirements and challenges of providing an always-on service. In LVS, a cluster of Linux servers appear as a single (virtual) server on a single IP address. Client applications interact with the cluster as if it were a single, high-performance, and highly-available server. Inside the virtual server, LVS directs incoming network connections to the different servers according to scheduling algorithms [1]. Scalability is achieved by transparently adding or removing nodes in the cluster. High availability is provided by detecting node or daemon failures and reconfiguring the system accordingly, on-the-fly.

C. DIFFERENT TECHNIQUE TO IMPLEMENT LVS

Linux Virtual Server (LVS) is an open source technology which can be used to construct a scalable and highly available server using a collection of real servers as shown in fig. 3.
LVS load balances a variety of network services among multiple machines by manipulating packets as they are processed by the Linux TCP/IP stack. One of the most common roles for LVS is to act as a front end to a farm of web servers.

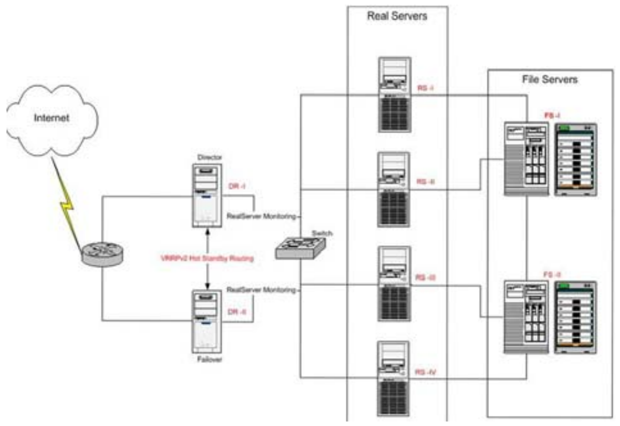

Fig. 2: High Availability Architecture

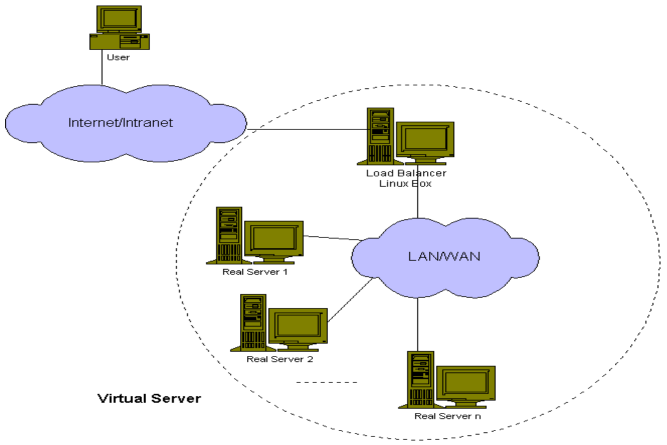

Fig. 3: Linux Virtual Server

Linux Virtual Server [1] is an enhancement to the Linux operating system which permits a set of servers to offer network services (e.g., ftp, http, ...) as a single virtual server. The aggregate capabilities of the individual hosts, or real servers, comprising the LVS cluster often exceeds those of any one of the constituent systems as a stand alone entity. Reliability is also increased because services are backed by multiple machines, thereby avoiding the single point of failure one would have with a single server.

IP load balancing techniques are quite scalable, and IPVS supports three different load balancing techniques: Linux Virtual Server via NAT (LVS/NAT), Linux Virtual Server via Tunneling (LVS/TUN), and Linux Virtual Server via Direct Routing (LVS/DR) [3].

D. VIRTUAL SERVER VIA NAT (VS/NAT)

Due to security considerations and the shortage of IP addresses in IPv4, more and more networks use private IP addresses that aren't allocated on the Internet. Network address translation is needed when hosts in internal networks want to access the Internet, or need to be accessed from the Internet. NAT can also be used to build a virtual server: parallel services at different IP addresses can appear as a virtual service on a single IP address. The architecture of Virtual Server via NAT is illustrated in fig. 4. The load balancer and real servers are interconnected by a switch or a hub.

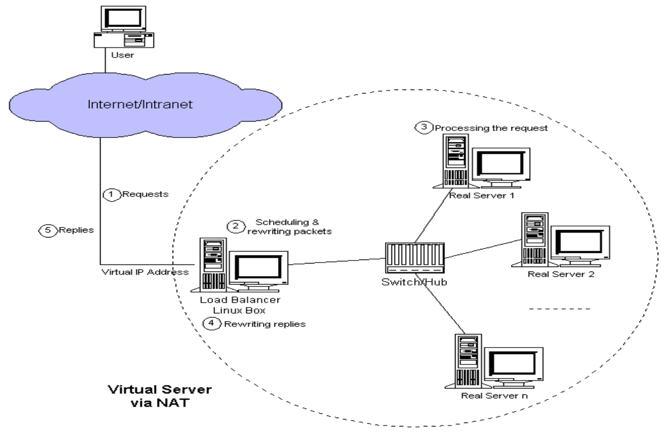

Fig. 4: Architecture of Virtual Server via NAT

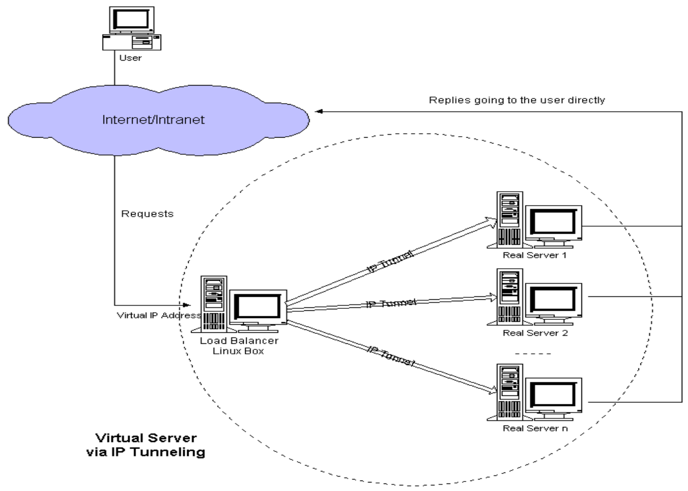

Fig. 5: Architecture of Virtual Server via IP Tunneling

E. VIRTUAL SERVER VIA IP TUNNELING (VS/TUN)

IP tunneling (also called IP encapsulation) is a technique to encapsulate IP datagrams within IP datagrams, which allows datagrams destined for one IP address to be wrapped and redirected to another IP address [2].

This technique can also be used to build a virtual server: the load balancer tunnels the request packets to the different servers; the servers process the requests, and return the results to the clients directly. Thus, the service appears as a virtual service on a single IP address. The architecture of Virtual Server via IP Tunneling is illustrated in fig. 5.

F. VIRTUAL SERVER VIA DIRECT ROUTING (VS/DR)

The VS/DR balancing approach is similar to the one implemented in IBM's Net Dispatcher [4]. The architecture of VS/DR is illustrated in Fig. 6.

The packet flow mechanism of VS/DR is similar to that of VS/NAT or VS/TUN. In VS/DR, the load balancer directly routes a packet to the selected server (the load balancer simply

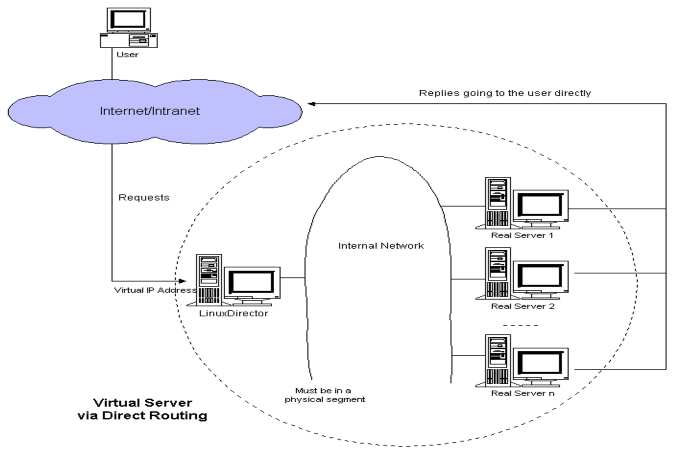

Fig. 6: Architecture of Virtual Server via Direct Routing VS/DR

changes the MAC address of the data frame to that of the server and retransmits it on the LAN). When the server receives the forwarded packet, the server determines that the packet is for the address on its loopback alias interface, processes the request, and finally returns the result directly to the user.

G. REASON FOR CHOOSING LVS-NAT USAGE SCENARIO

IP load balancing techniques are quite scalable, and IPVS supports three different load balancing techniques: Linux Virtual Server via NAT (LVS/NAT), Linux Virtual Server via Tunneling (LVS/TUN), and Linux Virtual Server via Direct Routing (LVS/DR). This research focuses on Linux Virtual Server via NAT (LVS/NAT). The main reason for choosing this type of load balancing technique is that in LVS/NAT the headers of the packets are overwritten by the director.

The director masquerades as the real servers. This creates the illusion that the real server is being contacted directly by the clients. The director must be the real servers default gateway for an LVS-NAT configuration to work properly [5]. As a result every response from a server is returned to the client via the director. Although this scheme is sub-optimal from performance perspective, it does have the benefit that the real server can run any operating system that supports TCP/IP. Another reason of choosing this technique over the other techniques is due to the advantage of any OS with TCP/IP support. Servers can use private IP addresses; only director needs public IP address.

III. METHODOLOGY

Fig. 7 shows the typical Linux Virtual Server configuration for the http service.

Any setup procedure (including the configuration script) will assume that the user have already setup all IP’s and network connections, apart from the Virtual IP and Director IP: i.e. setup Real Server IP1..n,. The configure script will setup the Virtual IP and the Director IP.

In a test setup, with all machines in the same computer using Vmware application, the user can make some virtual machines

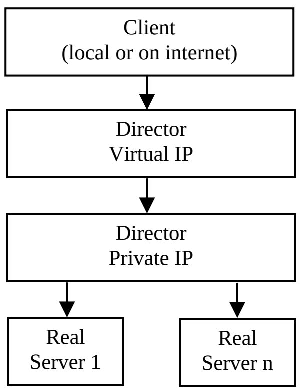

Fig. 7: Typical Linux Virtual Server configuration

on a single computer. The user will need a minimum of 3 virtual machines (1 client, 1 director, 1 realserver). If a user have 4 machines (i.e. 2 realservers), then user can see the load balancing (the connection from the client to the LVS being sent to one realserver, then the other). The user can setup an LVS with 2 virtual machines (1 client, 1 director) using the features of LVS, with the director also functioning as a real server.

The user needs following components:

A. CLIENT

Any machine, any OS, with a telnet client and/or an http client (e.g. Netscape, Lynx, Internet Explorer). In this setup WAPT is used for creating the client stress on the server.

B. DIRECTOR:

The director requires a machine running Pentium IV with 1GB RAM on 10Mpbs Ethernet with Linux Red Hat 9 with IPVS [3].

C. REAL-SERVER(S):

The real-server requires a machine running Pentium IV with 1GB RAM on 10Mpbs Ethernet with Linux Red Hat 9 with Apache web server [5].

IV. PERFORMANCE ANALYSIS

In order to observe the performance of the system, a differentiative analysis was done between the analytical results and empirical results; it requires increase the load and utilization of user and server bandwidth, the scenarios need to be of different combination of client load and iterations. For this each real server load with different user loads i-e 8, 16, 32 and 64 and different iterations i-e 1,5,10,15,20,25 and 30 were performed. The total number of connections can be calculated using the following formula.

$$ServedConnections = \sum_{iterations=n}^{client=m} mn(m+1)$$

Considering the ideal scenario where load is perfectly balanced by handle director server. We assume that, we have number of client's m and number of iterations n. so the total number of served connection can be computed.

A. SERVED RESULT

Let n=8 and m=1, the above equation (a) becomes, Served Connection = 36

B. SCENARIO 1

This scenario contains 8 clients with different iterations, who are working on real servers 1, 2 and 4 respectively. Fig. 7 shows the result of a scenario in which number of clients remains fixed i.e 8. These clients are increasing number of requests (iterations) to the director server. The director server is forwarding the requests to the real servers. The number of

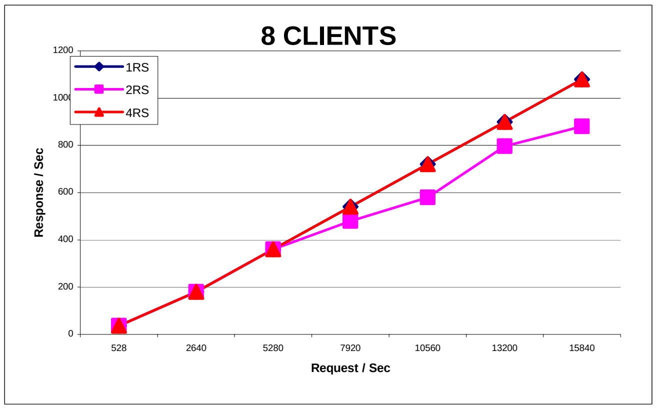

Fig. 7: 8 virtual user with increasing load and different 1, 2 and 4 real server(s)

real servers has been increased to see the proper effect of load balancing done by director server.

The results show that when a single real server is attached with director, all of the requests have been served by this single real server. The number of requests varies from 36 to 1080 and they are all served by the single real server. This shows a single real server (1 RS) appears to be performing in accordance to analytical results. The addition of a second real server (2 RS) provides a distribution of workload and approximately half of the requests are served by one real server and the remaining half by the other real server. The proper work load balancing by the director server can be observed through out the experiment results and it can be seen from table that all requests from clients are divided equally among real servers. The number of real servers serving the clients varies from 1 to 4.A single deviation from the ideal distribution curve can be observed when the number of requests has reached 30 by each of the 8 users with 2 real servers serving them. As mentioned above in the same chapter that simulations are run on a single machine with average computing resources. The abnormality can be explained as due to the use of same resources by other applications running on the same machine.

C. SCENARIO 2

In the above scenario the proper load balancing has been observed. In another variation number of clients has been increased to 16 and similarly the number of request generated by these clients increases linearly. The aim of this variation is to find out the saturation level up till when a single server can perform well and after need for work distribution is required. The results from Fig. 8 suggest that from 136 to 3400 requests generated by 16 users all of them are been served well by a single real server. But as the load increases and the number of requests from user's increases to 4080 a visible degradation is observed. This sudden change in performance has been properly investigated and it was found out that the director server was unable to handle so many requests and some of the requests were not forwarded to the real server. Among the request forwarded to the real server only 70% of over all request were served and other remained not served due to overload.

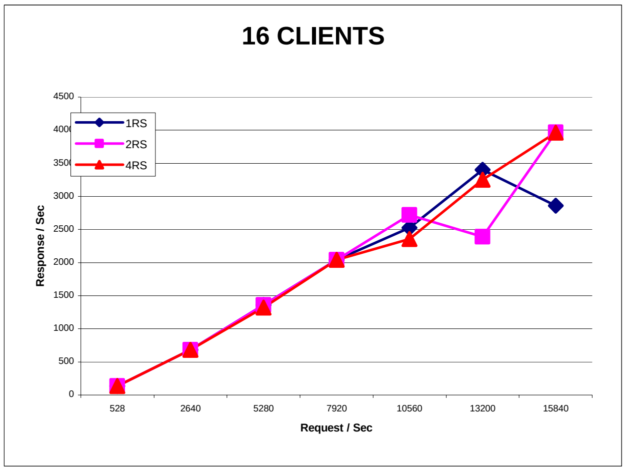

Fig. 8: 16 virtual user with increasing load and different 1, 2 and 4 real server(s)

More interesting observations are observed when the number of real servers increases. It has been observed that by increasing the number of real servers to 2 improves the over all percentage of job served by a single server by 27%. The overall number of jobs served reaches to 97% with two real servers. By increasing the number of real server it was required that the 100% job serving must be achieved, but in actual when the number of servers was further increased to 4. The performance approximately remains the same that is 97% of over all requests.

This can be concluded from the observations that further increase in real servers is waste of resources, and any addition in their number does not reflect any change in overall performance of the system.

D. SCENARIO 3

In the above scenario the proper load balancing has been observed. In another variation number of clients has been increased to 32 and similarly the number of request generated by these clients increases linearly. The aim of this variation is to find out the saturation level up till when a single server can perform well and after need for work distribution is required.

The results from Fig. 9 suggest that from 528 to 15840 requests generated by 32 clients all of them are been served well up to 3969 connections by a single real server at 10 iteration and remaining 1200 connections were lost. But as the load increases and the number of requests from user's increases to 15840 a visible degradation is observed. This sudden change in performance has been properly investigated and it was found out that the director server was unable to handle so many requests and some of the requests were not forwarded to the real server. More interesting observations are observed when the number of real servers increases. It has been observed that by increasing the number of real servers to 2 improves the over all percentage of job served by a single server is better in comparison of two real servers.This can be concluded from the observations that the director server has found to be the most crucial element in LVS.

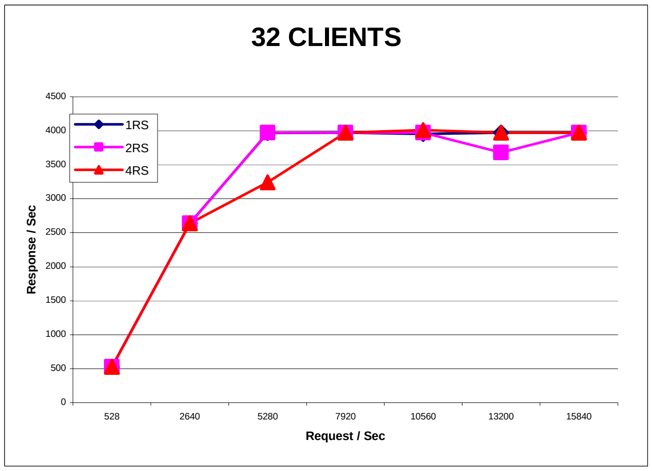

Fig. 9: 32 virtual user with increasing load and different 1, 2 and 4 real server(s)

The experimental results suggest that as the client load increases to a certain limit (approximately 4000 connection request in this case) the director starts to refuse new connections and doesn't forwards new request to real servers. This is recommended that a better configuration director server or redundant director server should be used to remove this bottle neck.

E. SCENARIO 4

As per above observations and before concluding the new things we increase the number of clients to 64 and similarly the number of request generated by these clients increases linearly.

The results from table 6.5 suggest that from 2080 to 62400 requests generated by 64 clients all of them are been served well approximately 4000 connections by a single real server at 5 and above iterations, remaining all connections were lost. But as the load increases and the number of requests from user's increases to 62400 a visible degradation is observed. This sudden change in performance has been properly investigated and it was found out that the director server was unable to handle so many requests and some of the equests

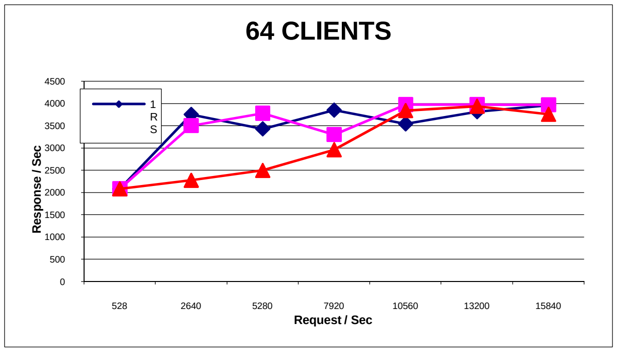

Fig. 10: 64 virtual user with increasing load and different 1, 2 and 4 real server(s)

were not forwarded to the real server. More interesting observations are observed when the number of real servers increases. It has been observed that no improvement is found to increase the number of real servers 2 to 4.

This can be concluded from the observations that the director server has found to be the most crucial element in LVS. The experimental results suggest that as the client load increases to a certain limit (approximately 4000 connection request in this case) the director starts to refuse new connections and doesn't forwards new request to real servers. This is recommended that a better configuration director server or redundant director server should be used to remove this bottle neck. This can also be concluded from the observations that further increase in real servers is waste of resources, and any addition in their number doest reflects any change in overall performance of the system.

V. CONCLUSION

The LVS has recently gained popularity due to its ability to provide high available services with load balancing and low cost features in cluster computing environment. One of the most valuable lessons to take away from this project is the ability to generate a high work load with a limit number of client machines. Our initial attempts were throttled by system resource limitation.

1. The director server has found to be the most crucial element in LVS. The experimental results suggest that as the client load increases to a certain limit (approximately 4000 connection request in this case) the director starts to refuse new connections and doesn't forwards new request to real servers. This is recommended that a better configuration director server or redundant director server should be used to remove this bottle neck.

2. In light of experimental results it is emphasized that the addition in number of real server should be sensible if real server has been increased without identifying the bottle neck at director server the increase will be waste of resources and invaluable.

3. The round robin scheduling algorithm is found to be efficient and appropriate under low load scenarios, but under high load conditions its performance is not found to be appropriate and other scheduling techniques should be applied.

A key asset of LVS is the willingness of the open source community to enhance and respond to the questions of users via the LVS mailing list. We have found it to be an invaluable resource that has assisted our efforts many times.

VI. FUTURE RECOMMENDATION

A significant number of LVS features are present like different job scheduling algorithms, different file system support and which can facilitate better performance and scalability analysis in addition to the analysis performed in this research. It would be interesting to see how LVS behaves under a more varied workload instead of the static data we used in this research. This would enable one to study the effects and benefits of the various of different protocols (e.g. HTTP, FTP etc.) and it may be interesting to see how well LVS can handle such workloads.

LVS-NAT is an important configuration because it allows yours real server to be platform independent, such flexibility would be important to anyone wishing to productize an LVS based solution so as to avoid making restrictions on the types of platforms which can function as servers. Other than LVS-NAT, two more approaches like LVS-DR and LVS-TUN exist and their performance analysis is needed to be done in comparison with LVS-NAT.

REFERENCES

[1] Haesun Shin, Sook-Heon Lee, and Myong-Soon: "Multicast-based Distributed LVS (MD-LVS) for improving scalability and availability", *ICPADS, Eighth International Conference on Parallel and Distributed Systems (ICPADS'01),* 2001, pp.0748-0754.

[2] Y. Saito, Bershad, B.N., and Levy, H.M, "Manageability, Availability and Performance in Porcupine: A Highly Scalable, Cluster-based Mail Service", *Proc. 16th ACM Symposium on Operating Systems Principles,* Charleston, December 1999.

[3] Wensong Zhang and Wenzhuo Zhang: "Linux Virtual Server Clusters: Build highly scalable and highly-available network services at low cost", *National Laboratory for Parallel and distributed processing* Changsha, Hunan 410073, China, November 2003.

[4] Wensong Zhang, Shiyao Jin, Quanyuan Wu: "Scaling Internet Services by Linux Director", *National Laboratory for Parallel & Distributed Processing* Changsha, Hunan 410073, China, *Proc. 4th International Conference on High Performance Computing in Asia Pacific Region 2000 IEEE,* vol.1, pp 176 - 183.

[5] Ibrahim Haddad and Greg Butler: "Experimental Studies of Scalability in Clustered Web Systems", *Proc. 18th International Parallel and Distributed Processing* Symposium *2004,* pp 185-193.