확장성과 고장 감내를 위한 효율적인 부하 분산기

김영환, 윤희용, 추현승
성균관 대학교 정보통신 공학부
yhkim@skku.ac.kr, {youn,choo}@ece.skku.ac.kr

Bi-active Load Balancer for enhancing of scalability and fault-tolerance of Cluster System

Young Hwan Kim, Hee Yong Youn, Hyun Seung Choo
School of Information and Communications
Sung Kyun Kwan University

Abstract

This paper describes the motivation, design and performance of bi-active Load balancer in Linux Virtual Server. The goal of bi-active Load balancer is to provide a framework to build highly scalable, fault-tolerant services using a large cluster of commodity servers. The TCP/IP stack of Linux Kernel is extended to support three IP load balancing techniques, which can make parallel services of different kinds of server clusters to appear as a service on a single IP address. Scalability is achieved by transparently adding or removing a node in the cluster, and high availability is provided by detecting node or daemon failures and reconfiguring the system appropriately. Extensive simulation reveals that the proposed approach improves the reply rate about 20% compared to earlier design.

Key word: cluster, load balancer, httperf, Linux Virtual Server

1. Introduction

With the explosive growth of the World Wide Web, some popular web sites are getting thousands of hits per second. As a result, the clients (browsers) experience slow response time and sometimes ever may not be able to access the web sites. Clustering with a single-system image view is the most commonly used approach solving this problem. With a server cluster, multiple servers behave as a single host from client's perspective.

There are two types of clustering architecture: centralized IP cluster and distributed IP cluster. Centralized IP cluster consists of one Load Balancer and several real servers. Load Balancer distributes incoming requests of clients to an appropriate real server based on load characteristics. The centralized IP cluster includes LVS (Linux Virtual Server) as S/W load balancing method, while Magic-Router, Local-Director and TCP Router are H/W load balancing methods. However, since this kind of solution creates a single-point-of-failure in the system, the availability is low. In addition, the scalability and fault-tolerance is low since the throughput of cluster is limited by the performance of Load balancer. The load Balancer bottleneck is more critical compared to the network bottleneck, and it limits the scalability of such servers in processing large number of simultaneous client requests. [1][2][5][7]

In order to solve this problem we propose bi-active that directs request packets to real servers including itself according to the scheduling algorithm and makes parallel services of the cluster to appear as a virtual service on a single IP address. This allows simulation highly scalable and fault-tolerance services. As a result, results show that the proposed scheme increases reply rate and decreases error rate. The proposed bi-active load balancer is kind of equalizer which is a server cluster software such as MPI, Ultramonkey allowing a user level process to intercept network packet.[1]

The rest of this paper is organized as follows: the next section gives a brief example of load balancers in Linux Virtual Server. Then Section 3 describes the proposed bi-active load balancer that we have implemented and tested. In Section 4 we show the performance of bi-active load balancer. Finally, in Section 5 we conclude the paper with a summary.

2. Previous Work

In the client/server applications, one end is the client and the other end is the server, and there may be a proxy in the middle. Based on this structure, there exist many ways to dispatch requests to a cluster of servers with different levels. Exiting request dispatching techniques are TCP Router and DPR (Distributed Packet Routing)[5]

2.1 TCP Router

TCP Router acts as a front-end that forwards requests for Web service to individual back-end servers of the cluster.

Two features of the TCP Router differentiate it from the Magic Router solution mentioned above. First, rewriting packets from server host kernels, which is not needed under the Magic-Router solution. Second, the TCP Router assigns connections to servers based on the state of the servers. This means that the TCP router must keep track of connection assignments. Figure 1 shows the structure of this system.

The architecture uses a TCP-based switching mechanism to implement a distributed proxy server. The motivation for this work is to address the performance limitations of client-side caching proxies by allowing a number of servers to act as a single proxy. The function of the server is similar to that of the Magic-Router. However, due to the caching functionality of the distributed proxy, additional issues are addressed mostly related to the maintenance of cache consistency among all servers in the cluster. If TCP router breaks down, whole system dose so. [5]

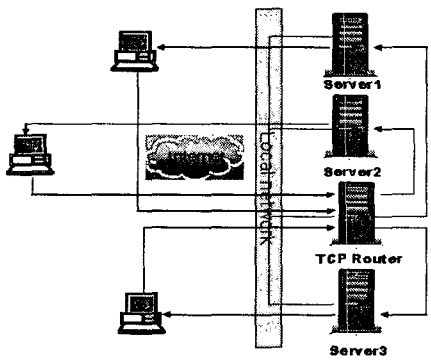

Figure 1 : TCP Router.

2.2 DPR (Distributed Packet Rewriting)

Under DPR the structure of any connection is conceptually a loop passing through three hosts (client and two server hosts). The entire set may have no host connected to all the servers. We refer to the first server host to which a packet arrives as the rewriter, and the second host as the destination.

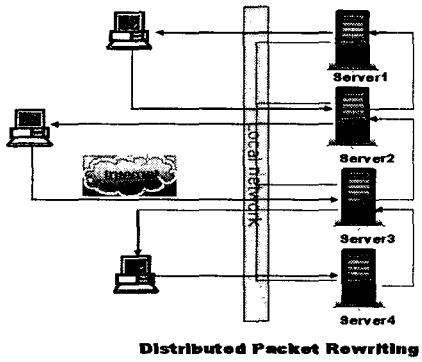

Figure 2 : DPR.

The DPR scheme assumes that requests arrive at the individual hosts of the server. This can occur in a number of ways. The simplest approach (which we currently use) is to distribute requests using Round-Robin DNS. Although requests may well arrive in a unbalanced manner because of the limitations of RR-DNS, The hosts experience balanced demands for service because of redistribution of the requests performed by DPR. As shown in Figure2.[5]

3. The Proposed Bi-active Load Balancer

We propose bi-active Load Balancer in the distributed packet LVS to enhance fault-tolerance and scalability, which combines the centralized IP cluster and the distributed IP cluster mechanism. The incoming packets from router are multicast to the Load Balancers. The bi-active Load Balancer achieves the load balancing using the Packet Accept Load Balancing Algorithm. The bi-active Load Balancer passes packets to real servers including itself and stand-by server for fault-tolerance. For example, if there are two load balancers and two real servers, four servers can service the clients.

Bi-active Load Balancer in the distributed packet LVS acts as follows: First, The load balancers have two Ethernet Cards, one for load balancer cluster, the other for real server cluster. Load balancers have real IP and virtual IP, and thus DR (Direct Routing) and NAT (Network Address Translation) method are used for LVS.

(a) Client sends request to the Apache Web server, which is set by Virtual IP address.
(b) The Ethernet addresses of Load balancers acting as host load balancers are selected by heartbeat daemon.
(c) After the Load balancer accepts the packet, it decides to process it or passes it to real server.
(d) If Load Balancer decides to accept the packets, Load Balancer directly starts to serve the client. If Load balancer distributes the packet to real server, real server starts to serve the client through load balancer using NAT (Network Address Translation) for packet forwarding. Figure 3 shows the proposed system with Bi-active Load Balance.

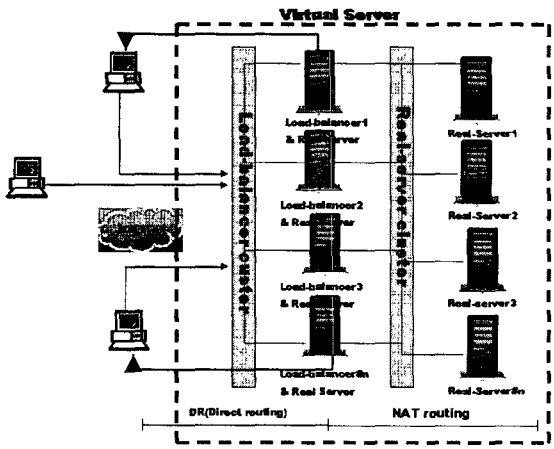

Figure 3 : Bi-active Load balancer.

In this mechanism, load balancer controls sub-clusters and distribute packets. In addition, load balancer acts as real

server if no packet comes in.

Bi-active Load balancers use DR if the distributed packet to load balancers, while NAT if the packet is distributed to real servers. As a result, the packets in load balancer can be processed faster than in real server.

The main advantage of the proposed mechanism is availability, fault tolerance, improved scalability, and easy implementation. The sub-clusters controlled by each load balancer are independent to each other. So, even though one of the Load Balancers fails, the others can continuously provide the service. As a result, the availability of the cluster system can be significantly increased. In addition, since a sub-cluster can be easily added as needed, the scalability is high.

4. Performance Evaluation

4.1 Software Configuration

4.1.1 Client Workload Generator

We use httperf[3][4], a tool for measuring web server performance. It provides a flexible facility for generating various HTTP workloads and measuring server performance. The focus of httperf is not on implementing one particular benchmark but on providing a robust, high-performance tool that facilitates the construction of both micro- and micro-level benchmarks. The three distinguishing characteristics of httperf are its robustness, which includes the ability to generate and sustain server overload, support for the HTTP/1.1 protocol, and its extensibility to new workload generators and performance measurements.

4.1.2 Testing Methodology

The objective of this section is to offer an accurate and clear understanding of how the experiment is conducted. Our primary goal is to stress the LVS (i.e. the bi-active load balancer node), and thus we keep our workload as simple as possible. The clients are assumed to generate requests based on the HTTP1.1 protocol. Each request is for the same web page to ensure that the web data would remain in the server's file cache, thereby eliminating any idiosyncrasies of file I/O on the web servers. The requested page is small enough (75.0 bytes) to be transmitted in a single packet without the need for fragmentation (given an MTU of 1500 bytes).

4.1.2.1 Apache Tuning

The Apache configuration file on each load balancer and real server is altered to disable logging and to keep a sufficient number of httpd daemons available to minimize overhead in responding to client requests.

4.1.2.2 Linux Tuning

The amount of socket memory on each system (client, load balancer and real server) is increased to 5Mbyte to allow a large TCP window size. In order to maximize the number of concurrent connections a given client could create ii was necessary to increase the number of per-process file descriptors as well as system wide limit on files and local port number.

4.1.2.3 LVS Configuration

All web servers are essentially of the same type of hardware, and thus we configure them with equal weight. All the clients request the same web content, and we employ round robin scheduling algorithm. In this LVS, we use ultra monkey 1.0.1 based on kernel 2.2 that creates load balance and highly available services on a local area network using open source components on the Linux operating system. There are important configuration items used for load balancer services. If we use two bi-active load balancers (one acts as host server, the others act as stand-by server), we need three IPs (one for virtual IP, and the others for bi-active load balancer). Hence we use IP alias supported by Linux Kernel 2.2.14. The actual configuration used is follows: [1][[5][6][7]

```
  #Setting IP alias ...
echo "Setting 203.252.46...."
/sbin/ifconfig lo 127.0.0.1
/sbin/ifconfig eth0 up
/sbin/ifconfig eth0 203.252.46.193
/sbin/ifconfig eth0:0 203.252.46.250
/sbin/ifconfig eth1 up
/sbin/ifconfig eth1 192.168.0.10
/sbin/ifconfig eth1:0 192.168.0.240
  #Setting IP route
echo "setting IProute....."
/sbin/route add -net 127.0.0.0
/sbin/route add -net 203.252.46.0 dev eth0
/sbin/route add -net 192.168.0.0 dev eth1
/sbin/route add -host 203.252.46.193 eth0
/sbin/route add -host 203.252.46.250 eth0:0
/sbin/route add -host 192.168.0.10 eth1
/sbin/route add -host 192.168.0.240 eth1:0
/sbin/route add default gw 203.252.46.1
```

4.2 Hardware Configuration

4.2.1 Bi-active Load balancer

The hardware used by the LVS load balancer was chosen because we felt it was representative of a typical system on the market today. The system has 256MB SDRAM, 700MHz Intel Pentium III CPU. The load balancer is equipped with two 3Com 3C905B-TXNM FAST ETHERLINK XL PCI Ethernet cards, each connected to 3Com 100MB switching HUB.

4.2.2 Real Servers

There are two real servers all of them contained a 500MHz Intel Pentium III CPU. Each system was booted with one 256MB of SDRAM and 3Com 3C905B-TXNM FAST ETHERLINK XL PCI Ethernet card connected to the 3Com 100 MB switching HUB.

4.2.3 Client

There are three sets of clients containing 500MHz Intel Pentium III CPU, 128MB of RAM and REALTEK RTL8139C 100MB Ethernet card.

4.3 Results

The results here are based on a series of test runs as shown in Table I. Some additional workloads were applied to some specific configurations in order to gain comprehensive understanding of a particular result.

Table I : LVS Test Matrix

Load balancer		Real Server		Config.	Client
Num.	Bi-active	Num.	used		Num.
1	O	1	O	NAT,DR	3(httperf)
1	X	1	O	NAT	3(httperf)
2	O	2	O	NAT,DR	3(httperf)
2	X	2	O	NAT	3(httperf)

As shown in Table I, We tested four cases, and the result shows that the proposed bi-active load balancer is more efficient than without it in terms of reply rate and error rate. Refer to Figure 4, 5, 6, and 7.

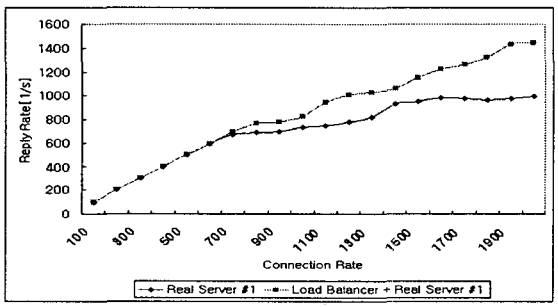

Figure 4: Reply rate with one bi-active Load balancer and real server Reply rate.

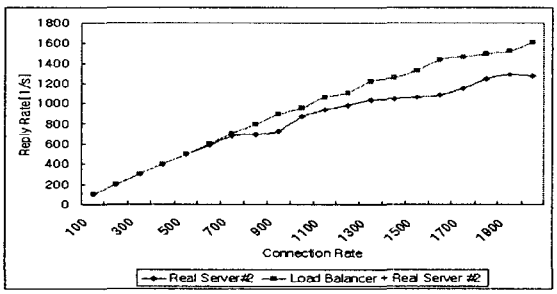

Figure 5 : Reply rate with two bi-active Load balancer and real server

Observe that reply rate increases up to 21.6% compared to existing approach.

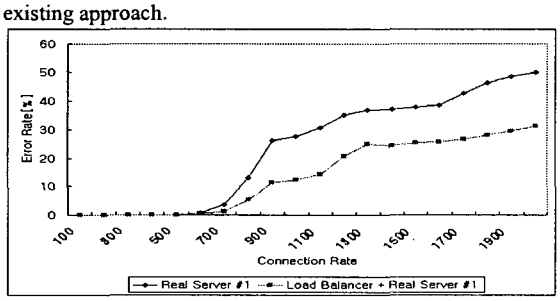

Figure 6 : Error rate with one bi-active Load balancer and real sever

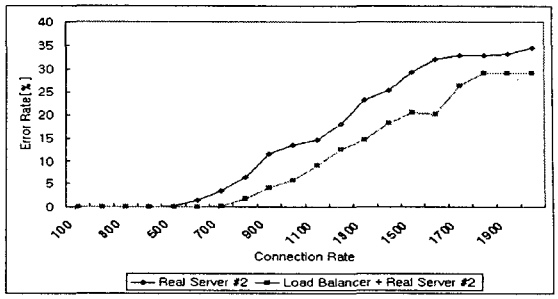

Figure 7: Error rate with two bi-active load balancer and real server

Figure 6 and 7 show that error rate decrease up to 18.9%.

4. Conclusion and Future Work

The proposed bi-active load balancer with Linux Virtual Server has been identified to be effective. One of the most valuable lessons is the ability to generate a high work load with a limited number of client machines. Our initial attempts were throttled by system resource limitations and thus we could not open more than 400 concurrent connections. After a substantial debugging effort, we were able to increase this to approximately 1500 concurrent connection.

A significant number of LVS features would benefit from more performance and scalability analysis in addition to the analysis we have done. It would be interesting to see how LVS behaves under a more varied workload instead of the static data we used in this report. This would enable one to study the effects and benefits of the various scheduling algorithms offered.

We skipped testing other scheduling methods (i.e. DR, Tunneling except for NAT). Perhaps LVS is more efficient with other scheduling methods. There is also a lack of standardization on testing and evaluating bi-active load balancer in general.

5. References

[1] Wensong Zhang. Linux Virtual Servers for Scalable Network Services.
http://www.linuxvirtualserver.org/ols/lvs.ps.gz

[2] Wensong Zhang. LVS Development page.
http://www.linuxvirtualserver.org/deployment.html

[3] Patrick O'Rourke, Mike Keefe: Performance Evaluation of Linux Virtual Server

[4] R. Fielding, J. Gettys, J. Mogul, H.Frystuk, and T. Berners-Lee.

Intenet Engineering Task Force, January 1997.

[5] Azer Bestavros, Mark Crovella, and Jun Liu.

[6] Linux IP Masquerade Resource.
See http://ipmasq.home.ml.org

[7] Wensong Zang, Shiyao jin and Qanyuan Wu, "Linux Virtual Server for Scalable Network Services" Ottawa Linux Symposium 2000, July 19-20nd, 2000