NetFlow-Based Approach to Compare the Load Balancing Algorithms

Chin-Yu Yang[1, 2] and Jian-Bo Chen[3]

[1]Dept. Computer Sci. & Information Eng., National Central University, Chung-Li, Taiwan, ROC
[2]Vanung University, Chung-Li, Taiwan, ROC
[3]Ming Chuan University, Taipei, Taiwan, ROC

Summary
The load balancing architecture is the most popular method to improve the performance of the server. The selection of the load balancing algorithms is one of the most important issues. In this paper, we use NetFlow to collect traffic for six load balancing algorithms, including least connections, round robin, minimum misses, hash, response time, and bandwidth. We compared their flow counts and packet counts separately. Both the WAN link load balancing and server load balancing are implemented. In addition, we also collected the burst traffic for server load balancing. The results for the performance of the algorithms are analyzed and compared.

Key words:
NetFlow, load balancing, cluster

1. Introduction

Clustering technologies enable incremental scaling of Internet server sites at modest cost. It is increasingly common in cluster-based service architectures to distribute incoming request traffic amount servers by using redirect switch. These server switches are called by various names including request distributors, front ends, redirectors, load balancers, network dispatchers, L4-L7 switches, interception switches, Web switches, content switches, and so on[1].

No matter which server switch we use, the most important issue is which load balancing algorithm are adopted. Different algorithms for load balancing architecture can be used to balance the load on a multi-server system based on the topology of the system[2]. The WAN link load balancing is used to balance the Internet connection for enterprise such as proxy server. The server load balancing is used to balance the load of server inside the enterprise such as web server. The purpose of these two systems is to alleviate the load of single server, but their topologies are quite different. In this paper, we use six different algorithms to analyze the behavior of the load balancing system. The six algorithms include least connections, round robin, minimum misses, hash, response time, and bandwidth. In order to avoid the influence of evaluating processes for the load balancing system, we use the port mirror function of the switch to export the traffic. We also use a probe to collect the traffic, then transfer the collected traffic into NetFlow format[3]. At last, we insert these data into SQL server for analyzing. Meanwhile, we also analyze some of the access logs from web servers and proxy servers[4, 5].

The rest of the paper is organized as follows. In section 2, we present the six load balancing algorithms. In section 3, we describe the system architecture. The experimental result is shown in section 4. The conclusion is described in section 5.

2 Load Balancing Algorithms

The load balancing architecture makes decisions regarding which server of a virtual server group to assign the new connection is based on the load balancing algorithms. The different algorithms operations available are as follows[6].

2.1 Least Connections

With the least connections algorithm, the number of connections currently opened on each backend server is measured in real-time. The backend server with least active connections is considered to be the best choice for the next client connection request. This algorithm is the most self-regulating, with the fastest servers typically getting the most connections over time.

2.2 Round Robin

With the Round-Robin algorithm, new connections are issued to each backend server in turn. That is, the first backend server in the group gets the first connection, the second backend server gets the next connection, followed by the third backend server, and so on. When all the backend servers in this group have received at least one connection, the process starts over with the first backend server.

Manuscript received October 5, 2008
Manuscript revised October 20, 2008

2.3 Minimum Misses

The minimum misses algorithm is optimized for WAN link load balancing. It uses IP address information in the client request to select a server. The specific IP address information used depends on the application.

For WAN link load balancing, the client destination IP address is used. All requests for a specific IP destination address are sent to the same server. This algorithm is particularly useful in caching applications, helping to maximize successful cache hits. Best statistical load balancing is achieved when the client IP addresses are spread across a broad range of IP subnets.

For server load balancing, the client source IP address and backend server IP address are used. All requests from a specific client are sent to the same backend server. This algorithm is useful for applications where client information must be retained on the server between sessions. With this algorithm, backend server loading becomes most evenly balanced as the number of active clients with different source or destination addresses increases.

2.4 Hash

The hash algorithm uses IP address information in the client request to select a backend server. The specific IP address information used depends on the application. For WAN link load balancing, the client destination IP address is used. All requests for a specific IP destination address will be sent to the same server. This is particularly useful for maximizing successful cache hits.

For server load balancing, the client IP address is used. All requests from a specific client will be sent to the same backend server. This option is useful for applications where client information must be retained between sessions.

When selecting a backend server, a mathematical hash of the relevant IP address information is used as an index into the list of currently available servers. Any given IP address information will always have the same hash result, providing natural persistence, as long as the backend server list is stable. However, if a server is added to or leaves the system, then a different backend server might be assigned to a subsequent session with the same IP address information even though the original server is still available. Open connections are not cleared.

The hash algorithm provides more distributed load balancing than minimum misses at any given instant. It should be used if the statistical load balancing achieved using minimum misses is not as optimal as desired. If the load balancing statistics with minimum misses indicate that one backend server is processing significantly more requests over time than other servers, consider using the hash algorithm.

2.5 Response Time

The response time algorithm uses backend server response time to assign sessions to servers. The response time between the servers and the load balancer is used as the weighting factor. The load balancer monitors and records the amount of time it takes for each backend server to reply to a health check to adjust the backend server weights. The weights are adjusted so they are inversely proportional to a moving average of response time. In such a scenario, a server with half the response time as another server will receive a weight twice as large.

2.6 Bandwidth

The bandwidth algorithm uses backend server octet counts to assign sessions to a server. The load balancer monitors the number of octets been sent between the server and itself. Then, the weights of backend server are adjusted so they are inversely proportional to the number of octets that the backend server processed during the last interval.

Backend servers that process more octets are considered to have less available bandwidth than those that have processed fewer octets. For example, the backend server that processes half the amount of octets over the last interval receives twice the weight of the other backend servers. The higher the bandwidth used, the smaller the weight assigned to the server. Based on this weighting, the subsequent requests go to the backend server with the highest amount of free bandwidth. These weights are automatically assigned.

3 System Architecture

3.1 NetFlow Traffic Collections

The architecture of traffic collection is shown in Fig. 3. The server switch[7] acts as a virtual front-end processor to clusters of real servers connected via direct attachment to switch ports or indirectly through hubs and switches.

In network environments, NetFlow is probably the most useful standard for network traffic accounting. In our experiment, we use both a NetFlow v5 probe (nProbe) [8] and collector to monitor the flows within the server switch.

When the nProbe activated, nProbe will collect traffic data and emit NetFlow v9 flows towards the specified collector. A set of packets with the same (src ip & port, dst ip & port, protocol #) is called flow. Every flow, even a very long-standing ISO CD image download, has a limited lifetime; this is because the flow collector should periodically receive flow chunks for accounting traffic precisely. Then we use a collector to receive the NetFlow v9 flows, and store all the information into MySQL database.

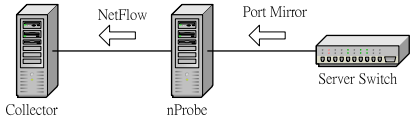

Figure 1. NetFlow traffic collections

3.2 WAN Link Load Balancing Architecture

The WAN link load balancing architecture is to share the load for WAN link, especially for the HTTP traffic. In order to improve the performance, we use four proxy servers instead of the main proxy server. The IP address of the main proxy server was configured in the Server switch. Every time when a client makes a request to the main proxy server, the Server switch will choose one proxy server to process the request by its algorithm. The four proxy servers are all installed with two network interface cards. One NIC binds the local IP address in our environments and the other NIC binds the IP address given by the ADSL ISP provider as shown in Fig. 2.

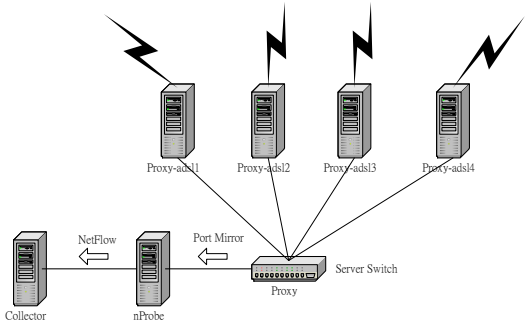

Figure 2. Proxy server architecture

3.3 Server Load Balancing Architecture

The server load balancing architecture is to distribute the client requests to several WWW servers. In order to improve the performance, we use eight WWW servers instead of the main WWW server. The IP address of the main WWW server was configured in the Server switch. Every time when a client makes a request to the WWW server, the Server switch will choose one WWW server to serve it by the pre-defined algorithm. The eight WWW servers share the same contents. The architecture is shown in Fig.3.

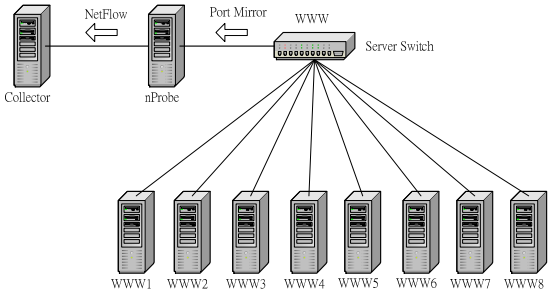

Figure 3. WWW server architecture

4. Experimental Results

In our experiments, we collected the NetFlow data for Proxy servers and WWW servers. The flow counts and packet counts are presented. We also calculated the standard derivation for each load balancing algorithm. The burst traffics for WWW were also collected.

4.1 Proxy server load balancing

We used the six different load balancing algorithms to collect the traffic data. Each algorithm collects traffic about 24 hours. The data we collected are shown in Table 1. The standard derivations of the data are shown in Table 2 and Fig. 4 and 5.

Table 1. Proxy server load balancing traffic data

Algorithms		Proxy-adsl1	Proxy-adsl2	Proxy-adsl3	Proxy-adsl4
Hash	Flow	256998	248630	233336	237907
	Packet	2983279	1880752	2179256	2082662
RoundRobin	Flow	261674	251490	243100	262147
	Packet	3441868	1980849	2226754	2022514
Least Conns	Flow	274839	275487	241918	261488
	Packet	3409218	2204512	2537912	2156230
Min Misses	Flow	264197	269308	246870	264323
	Packet	3104660	2093349	2351239	2176019

Res-ponse	Flow	253151	340339	312145	275066
	Packet	3876566	2671434	2630859	2050195
Band-width	Flow	244235	231005	256055	231851
	Packet	3190323	1814767	2412468	1828310

Table 2. Proxy server load balancing standard derivations

Algorithms	Flow	Packet
Hash	10662	484108
Round Robin	9109	690987
Least Conns	15728	580204
Min Misses	9829	461592
Response	38722	767274
Bandwidth	11840	648779

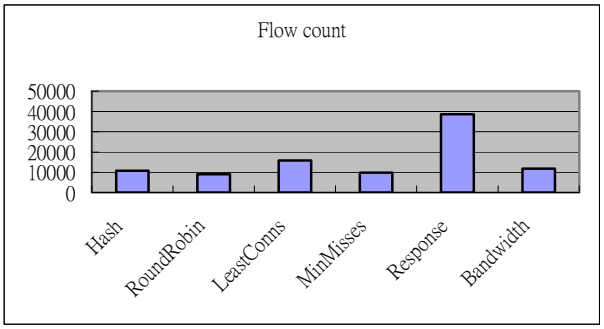

Figure 4. Standard derivations for flow counts

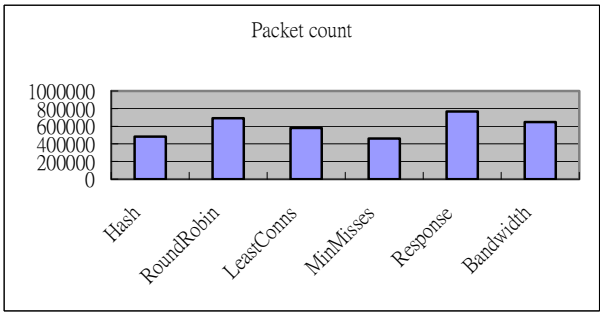

Figure 5. Standard derivations for packet counts

4.2 WWW server load balancing

For server load balancing, we collected the data and calculated the standard derivations of the data Table 3 and Fig. 6 and Fig. 7.

Table 3. WWW server load balancing standard derivations

Algorithms	Flow	Packet
Hash	3900	19522
RoundRobin	4524	16532
LeastConns	3656	14180
MinMisses	1516	19046
Response	2171	38197
Bandwidth	398	42490

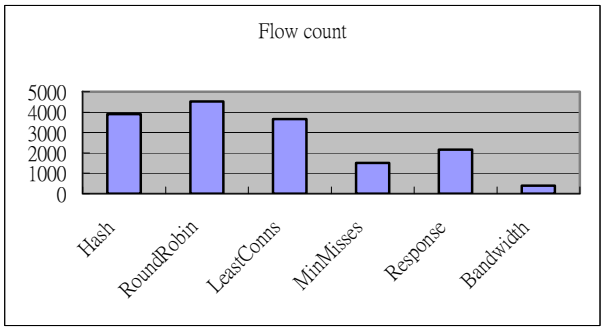

Figure 6. Standard derivations for flow counts

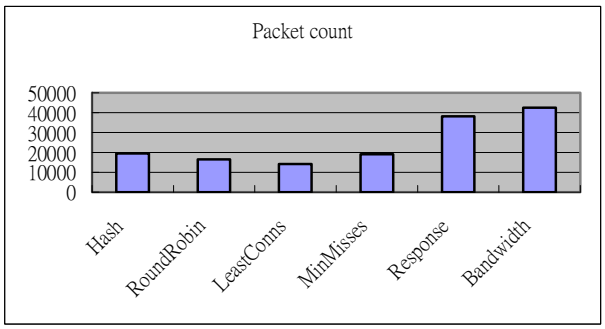

Figure 7. Standard derivations for packet counts

4.3 Burst traffic of WWW server load balancing

In this experiment, we want to know what happenes when the burst traffic occurs. We collected three period of time with burst traffic, and with each period, we collected ten minutes. But due to the session problems of the WWW

servers, we only tested the hash algorithm. The data we collected are shown in Table 4 and Fig. 8 and Fig. 9.

Table 4. WWW server load balancing with brust data

Servers	Time slot 1		Time slot 2		Time slot 3	
	Flow	Pack	Flow	Pack	Flow	Pack
WWW1	5473	3617	8473	4867	6998	6127
WWW2	9369	5012	1047	5421	1458	8711
WWW3	5190	4070	5737	4402	7801	6724
WWW4	6789	4995	5910	4310	7333	8382
WWW5	1695	7971	1023	5428	9540	7059
WWW6	8344	4635	7364	4705	8029	5990
WWW7	7222	4579	9065	5346	8625	6393
WWW8	5954	4517	6717	5093	6510	5659

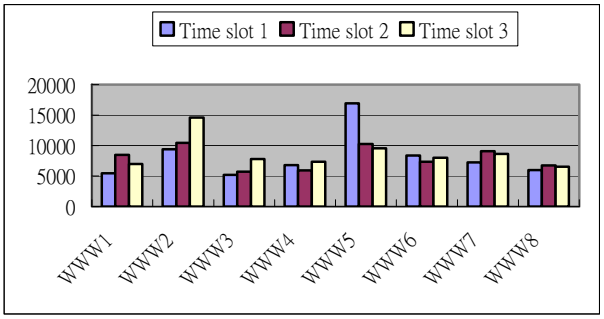

Figure 8. The flow counts statistics

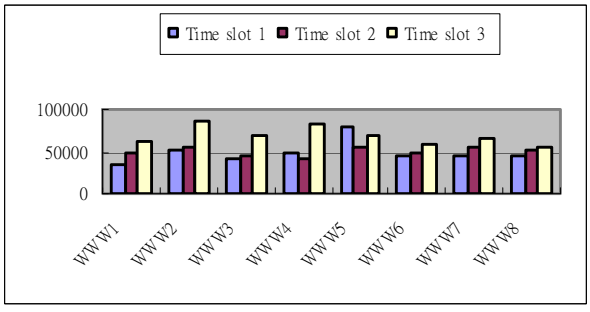

Figure 9. The packet counts statistics

4.4 Analysis of Experimental Results

It seems that the flow count for each server must be the same when we use round robin as the load balancing algorithm. But the results showed that the standard derivation of round robin is not approximately zero. The reason is that the NetFlow collector reports the collected data periodically. When a flow, or a connection, transfers a large amount of data, NetFlow will spilt this one flow into several flows, so the statistic data will not be the same for each server.

The minimum misses algorithm for the WAN link load balancing, or proxy server, will get the highest performance because of its high hit rate. The standard derivation cannot show this advantage because the high hit rate will reduce the response time, not dispatch the traffic fairly.

5. Conclusion

In this paper, we used the NetFlow to collect the traffic data of different load balancing algorithms. We can find out that each algorithm in the different kinds of traffic have different performance. We use the flow counts and packet counts to verify the load balancing of each algorithm. As far as we know, these algorithms cannot 100% balancing the loads because the unexpected network conditions and server loads.

References

[1] Jeffery S. Chase, “Server Switching: Yesterday and Tomorrow,” Proc. The Second IEEE Workshop on Internet Applications, 2001
[2] Hemant B. More & Jie Wu, “Throughput Improvement Through Dynamic Load Balance,” Proc. IEEE SOUTHEASTCON. Bibliographic Details, 1994
[3] NetFlow Services Export Version 9, RFC 3954
[4] Lili Qiu, Venkata N. Padmanabhan, Geoffery M.Voelker, “On the Placement of Web Server Replicas,” Proc. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2001
[5] Scot Hull, “Content Delivery Networks: Web Switching for Security, Availability, and Speed, “McGraw Hill, 2002.
[6] Notel Networks, Alteon Link Optimizer Application Guide, Release 1.0, 2002
[7] http://www.nortelnetworks.com
[8] http://www.ntop.org/nProbe.html

Chin-Yu Yang received the MS degree in department of electrical engineering in Yuan Ze University, Jhongli, Taiwan, Republic of China, in 1997, and he is a PhD student in the department of computer science and information engineering in National Central University, Jhongli,

Taiwan, Republic of China. He is currently an instructor in the department of computer science and information engineering in Vanung University, Jhongli, Taiwan, Republic of China. His research interests include wireless network, network security and network management.

Jian-Bo Chen received the MS degree in the department of electrical engineering in National Taiwan University, Taipei, Taiwan, Republic of China, in 1995, and PhD degree in the department of computer science and engineering in Tatung University, Taipei, Taiwan, Republic of China, in 2008. He is currently an assistant professor in the department of information and telecommunications engineering in Ming Chuan University, Taoyuan, Taiwan, Republic of China. His research interests include network management, network security, and load balance.