The Three-level Approaches for Differentiated Service in Clustering Web Server*

Myung-Sub Lee and Chang-Hyeon Park

School of Computer Science and Electrical Engineering, Yeungnam University
Kyungsan, Kyungbuk 712-749, Republic of Korea
{skydream, park}@yu.ac.kr

Abstract. This paper presents three-level approaches for the differentiated Web QoS. A kernel-level approach adds a realtime scheduler to the operating system kernel to keep the priority of the user requests determined by the scheduler in the Web server. An application-level approach which uses IP-level masquerading and tunneling technology improves the reliability and response speed of the Web services. A dynamic load-balancing approach uses the parameters related to the MIB-II of SNMP and the parameters related to the load of the system resources such as memory and CPU to perform load balancing dynamically. These approaches proposed in this paper are implemented using a Linux kernel 2.4.7 and tested in three different situations. The result of tests shows that the appraches support the differentiated services in clustering web server environment.

Keywords: Differentiated QoS, Dynamic load balancing, SNMP, MIB-II, Realtime scheduler

1 Introduction

Recently the technologies related to Web QoS(Quality of Service) which guarantees the quality of Web services are becoming more important[1,2,3]. Particularly for the differentiated quality of Web services, Web servers are required to be able to classify contents depending on the importance of the information and the priority of the customer and perform scheduling among the classified contents. However, most Web servers currently provide best effort services on a FIFO(First In First Out) basis only. This means that, when they are overloaded, the servers cannot provide the appropriate services for the premium users[4,5].

Hence, a new server model is needed so that it may guarantee the quality of services by classifying services according to specific criteria and providing differentiated services. Despite the rapid expansion in Web use, the capacity of current Web servers is unable to satisfy the increasing demands. Consequently,

* This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

even if a Web server providing differentiated services is developed, it cannot guarantee perfect service.

As a resolution for Web QoS, Web server technologies employing load balancing have been proposed. However, the exiting load balancing technologies for Web servers still have some problems, such as incompatibility between different client application programs[6], inability to process overloaded servers[7], overload when processing HTTP requests/replies[8, 9, 10, 11], packet conversion overheads[12, 13], and etc.

This paper proposes three-level approaches for implementing load balancing Web servers that can guarantee differentiated Web QoS. In the first approach, a scheduling module is added to Web server, which assigns a priority to a client request according to its importance, and a realtime scheduler is inserted into the OS kernel so that the assigned priority can be kept in the OS, and thereby more efficient differentiated service is provided. In the second approach, the load balancing Web server is configured using masquerading and tunneling technologies to distribute the load by class, thereby the reliability and response time of the Web services are improved. The third approach uses the parameters related to the MIB-II of SNMP and the parameters related to the load of the system resources such as memory and CPU to perform load balancing dynamically.

2 A Differentiated Web Service System

The proposed system uses three-level approaches: kernel-level approach, application-level approach, and load-balancing approach.

2.1 Kernel-level approach

For the client requests, this approach maintains their priority order determined by the Web server in the OS kernel. This approach is implemented by mapping the scheduling processes in the Apache Web server to the realtime scheduling processes in the OS kernel.

When the client requests come through a Network Interface Card(NIC), the Web server receives them from port 80 in the TCP listening buffer, classifies them by connection according to specific classification policies(client IP, URL, file name, directory, user authentication, etc.), assigns the proper priority, then inserts them into the corresponding queues. Thereafter, at the same time the requests are being scheduled, the scheduling processes in the Web server are mapped one-to-one to the processes in the realtime scheduler(Montavista in this paper) in the Linux OS kernel.

2.2 Application-level Approach

The load balancing Web server proposed in this paper has a high performance and expansibility by enhancing the packet transmission rate and by resolving the bottleneck in the load balancer through the use of IP-level masquerading and

tunneling. In the proposed system, a single load-balancer distributes the requests to several real servers, which share a common IP address, using a masquerading technique so that they look like a single server from the outside. IP masquerading hides the real servers behind a virtual server that acts as a gateway to external networks.

2.3 Dynamic load balancing Approach

The load balancer analyzes the load of the actual server, by analyzing the utilization of the ethernet and the rate of systemic load, after processing the MIB-II value that is related to the load of SNMPv2.

The systemic load analysis proposed in this paper is the value in which the utilization of the system is added to all the utilization of the ethernet. The equation (1) for load analysis is as follows:

$$Total\_load = Ethernet\_Utilization + Sys\_Utilization \tag{1}$$

The ethernet utilization, given equation (2), means all the traffic amount of input and output of the load balancer. In other words, the sum of all the bit number of packets that transmitted from the sending side and all the bit number that the receiving side received is divided by the whole bandwidth of the network. The variables used to measure the utilization of the ethernet in this paper are listed in table 1.

$$Ethernet\_Utilization = (total\_bit\_sent + total\_bit\_received)/bandwidth \tag{2}$$

Table 1. The variables of ethernet utilization

Item	Explanation
x	Previous polling time
T	Polling interval
ifInOctets	The number of received octets
ifOutOctets	The number of transmitted octets
sysUpTime	System boot time
ifSpeed	Current network bandwidth

In order to determine the utilization of an ethernet network, the input and output traffic must be added, and then the sum is to be divided by the maximum transmission speed. The ethernet traffic analysis equation is given by equation (3).

$$\frac{(ifInOctets_{(x+t)} - ifInOctets_{(x)} + ifOutOctets_{(x+t)} - ifOutOctets_{(x)}) \times 8}{(sysUpTime_{(x+t)} - sysUpTime_{(x)}) \times ifSpeed \times 10} \times 100 \tag{3}$$

The utilization of the system is the sum of memory utilization, CPU average utilization, and disc utilization, as shown in equation (4). The variables used to measure the utilization of the system in this paper are listed in table 2.

$$Sys\_Utilization = memSwapLoad + laLoad + dskLoad \tag{4}$$

Table 2. The variable of system utilization

Item	Explanation
memTotalSwap	The total space of swap memory
memAvailSwap	The available space of swap memory
memTotalReal	The total space of physical memory
memAvailReal	The available space of physical memory
memTotalFree	The total space of free memory
laLoad_x	The average load of CPU for x minutes
dskTotal	The total disk space
dskAvail	The available disk space
dskUsed	The used disk space

In order to determine the utilization of a system, the memory utilization and CPU utilization and disk utilization must be added. The calculation equation of system utilization is given below:

$$memSwapLoad = \frac{memTotalSwap - memAvailSwap}{memTotalSwap} \times 100 \tag{5}$$

$$memRealLoad = \frac{memTotalReal - memAvailReal}{memTotalReal} \times 100 \tag{6}$$

$$laLoad = laLoad\_x \times 100 \tag{7}$$

$$dskLoad = \frac{dskUsed}{dskTotal} \times 100 \tag{8}$$

Equation (5) is used to calculate the use rate of swap memory using the *memTotalSwap* value and the *memAvailSwap* value. Equation (6) is to calculate the utilization of physical memory using the *memTotalReal* value and the *memAvailReal* value. Equation (7) is to calculate the average utilization of CPU for x minutes by percent. Equation (8) is to calculate the utilization of disc using the *dskTotal* value and the *dskUsed* value.

3 Implementation and Experiment

The differentiated Web service system proposed in this paper is implemented using a Linux Kernel 2.4.7 and PCs with a Pentium-III 800MHz processor and a 256MB RAM, while the test environment is built by networking three clients, one

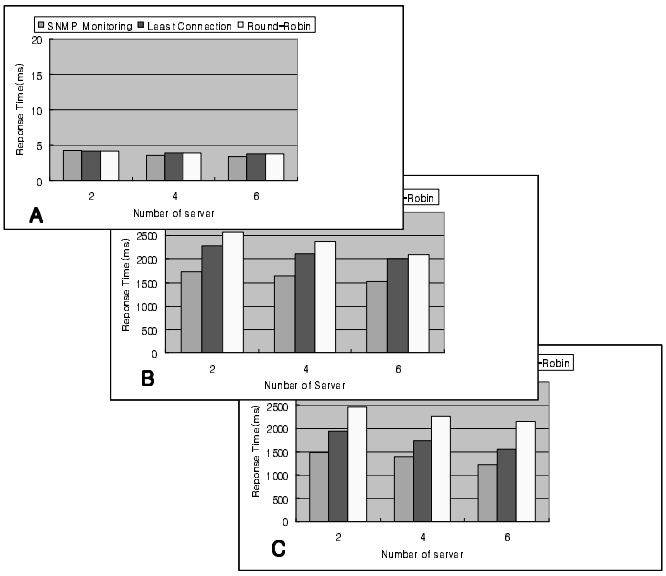

Fig. 1. Experimental graphs of the ethernet and system utilization

load balancer, two servers, and one monitoring server. An Apache Web Server 2.4.17 is modified for the Web server, and a Montavista realtime scheduler is added to the Linux kernel.

In this paper, HP's *httperf* program, and *AB*(Apache HTTP server Benchmark tool) that measure the response speed of the Apache server are used to evaluate the capability of the Web server.

Tests are carried out for three cases: when the servers are not overloaded(test 1), when the servers are overloaded(test 2), and when the servers are overloaded and some requests are subsequently stopped(test 3). In test 1, the virtual IP address is 165.229.192.14, the total number of connections 50000, the number of concurrent users per session 1, and the number of calls per session 50.

Fig. 1(A) presents the results of the ethernet and system utilization, which shows the reply changes of Web servers upon the three clients. As the servers are not overloaded, the graphs are almost the same. However, if Web servers are overloaded, In Fig. 1(B), the system proposed in this paper realized better capability 1.3 times better than the least connection scheduling, and 1.5 times better than the round-robin scheduling. In test 3, which uses the same conditions as test 2. But the script code was formatted with such mechanism that the CPU load of real server 1 would increase. As shown in Fig. 1(C), the proposed mechanism realized 1.3 - 1.6 times better capability than other scheduling algorithms. It was because load balancing was precise owing to the periodical measure of present load of every real server.

4 Conclusion

To implement a differentiated Web service system that provides differentiated services according to information importance or user priority, this paper proposed three-level approaches: a kernel-level approach, an application-level approach and a dynamic load-balancing approach. In the kernel-level approach, a realtime scheduler is added to the kernel, while in the application-level approach, the load balancer is implemented using an IP-level masquerading technique and tunneling technique. The performance of the load balancing system was tested in three different situations, and the results confirmed that the system supported differentiated Web services.

References

1. R.Fielding, J. Getys, J. Mogul, H. Frystyk, and T. Berners-Lee, Hypertext Transfer Protocol HTTP/1.1, IETF (1997)
2. N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating User Perceived Quality into Web Server Design", Proc. of the 9th International World Wide Web Conference, Amsterdam, Netherlands (2000) 92-115
3. N. Vasiliou and H. Lutfiyya., "Providing a Differentiated Quality of Service in a World Wide Web Server", Proc. of the Performance and Architecture of Web Servers Workshop, Santa Clara, California USA (2000) 14-20
4. Apache Group, http://www.apache.org/.
5. R. Bhatti and R. Friedrich, "Web Server Support for Tiered Services", IEEE Network (1999) 64-71
6. Chad Yoshikawa, Brent Chun, Paul Eastharn, Armin Vahdat, Thomas Anderson, and David Culler, "Using Smart Clients to Build Scalable Services", USENIX'97, http://now.cs.berkeley.edu/ (1997)
7. Thomas T. Kwan, Robert E. McGrath, and Daniel A. Reed, "NCSA's World Wide Web Server: Design and Performance", IEEE Computer (1995) 68-74
8. A. Dahlin, M. Froberg, J. Walerud and P. Winroth, "EDDIE: A Robust and Scalable Internet Server", http://www.eddieware.org/ (1998)
9. Ralf S.Engelschall, "Load Balancing Your Web Site: Practical Approaches for Distributing HTTP Traffic", Web Techniques Magazine **3** http://www.webtechniques.com (1998)
10. Edward Walker, "pWEB - A Parallel Web Server Harness", http://www.ihpc.nus.edu.sg/STAFF/edward/pweb.html (1997)
11. Daniel Andresen, Tao Yang, Oscar H. Ibarra, "Towards a Scalable Distributed WWW Server on Workstation Clusters", Proc. of 10th IEEE Intl. Symp. of Parallel Processing(IPPS'96) (1996) 850-856
12. Eric Anderson, Dave Patterson, and Eric Brewer, "The Magicrouter: an Application of Fast Packet Interposing", http://www.cs.berkeley.edu/∼eanders/magicrouter/ (1996)
13. Wensong Zhang, "Linux Virtual Server Project", http://proxy.iinchina.net/∼wensong/ippfvs (1998)
14. Montavista Software, http://www.montavista.com/.