제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

리얼 서버의 실시간 서버 모니터에 의한 최적 로드 밸런싱 알고리즘에 관한 연구

한일석*, 김완용*, 김학배*
*연세대학교 전기전자공학과
E-mail: hbkim@yonsei.ac.kr

A Study for an Optimal Load Balancing Algorithm based on the Real-Time Server Monitor of a Real Server

*Ilseok Han, *Wanyong Kim and *Hagbae Kim
*Dept. of Electrical and Electronic Eng, Yonsei University, Korea

Abstract

As a consequence of WWW large popularity, the Internet has suffered from various performance problems, such as network congestion and overloaded servers. These days, it is not uncommon to find servers refusing connections because they are overloaded. Web server performance has always been a key issue in the design and operation of on-line systems. With regard to Internet, performance is also critical, because users want fast and easy access to all objects (e.g., documents, graphics, audio, and video) available on the net. To solve this problem, a number of companies are exploring the benefits of having multiple geographically or locally distributed Internet sites. This requires a comprehensive scheme for traffic management, which includes the principle of an optimal load balancing of client requests across multiple clusters of real servers. This paper focuses on the performance analysis of Web server and we apply these results to load balancing in clustering web server. It also discusses the main steps needed to carry out a WWW performance analysis effort and shows relations between the workload characteristics and system resource usage. Also, we will introduce an optimal load balancing algorithm base on the RTSM (Real-Time Server Monitor) and Fuzzy Inference Engine for the local status of a real server, and the benefits is provided with of the suggested method.

I. INTRODUCTION

Load balancing is a method of distributing the web traffic across an array of real servers, in order to help enhancing the performance and availability of web server. By distributing user requests across an array of real servers, effective load balancing solutions can relieve network traffic overload, reduce the burden of web server and improve the browsing performance for users. As Internet sites begin to handle more traffic and services support protocols that are more complex, availability and fault tolerance become critical. Every transaction and user interactions should be completely reliable to maintain optimal server Quality of Service.

So, a number of researchers developed load balancing algorithms, especially using fuzzy logic, effectively to reflect real server's global uncertainty [1-3]. However, most conventional fuzzy approaches for load balancing were merely send simple information about current available resources of real servers and network traffics to the load balancer, which determined the service priorities based on some primitive a real server information and performed traffic distributions. Thus, those approaches may cause serious local network bottlenecks because of the overheads incurred by frequent information transactions between the load balancer and real servers. Also, because the processing capacity of real servers, H/W and S/W specifications of real servers differs, it can not perform a suitable traffic distribution through the special quality of real servers, efficiently. So, we analyze the performance metrics of real servers and will go to apply the load balancing scheduling policy.

This paper proposes both an optimal load-balancing algorithm and a novel way to an accurately analyze the local status and the performance metrics (CPU utilization, the amount of memory usage, the amount of network traffics, throughput, load average and etc.,) of a real server through the ANFIS (Adaptive Network based Fuzzy Inference System) modeling.

II. THE PROPOSED ALGORITHMS

For a clustering web servers and user-part network in home network environments, the key factor for maximizing both the performance and the dependability (availability and/or reliability) would be a load balancing technique in the server that provides certain services for either a number of users or a variety of home appliances connected to the gateways. The load balancer efficiently distributes network traffics among real servers to avoid overburdened status of individual servers. For enhancing the accessibility to all resources, the load balancer should be able to analyze the traffic of real servers as accurately as possible. Therefore, state analysis about real servers is a major criterion for problem determining the effective traffic distribution.

Most of a current load-balancing algorithm analyzes logs files one-by-one, creating the statistics in a linear fashion. This can cause some problems when handling load balanced web servers (real servers). Load balanced real servers are used to increase the performance of web sites by spreading the load of the traffic across several real servers, each typically having its own copy of the web site. This set up is completely transparent to the visitor of the site; it means that the client cannot notice that each page he/she views may come from a different physical a real server. However, each of real servers also creates a separate set of log files which must be perfectly analyzed together to generate correct statistics. This is especially important for user and the current available resource calculations when many requests can be part of the same user. In this scenario, the requests are spread out among some different real servers, not until you look at all the log files together and you get the complete picture.

Most of a current load-balancing algorithms read the log files one-by-one they cannot track users that span several different log files. To solve this ineffective situation, the proposed algorithm can merge the separate log files, one for each a real server, into one file where every request is organized in order of time stamps on the load balancer. This file can, then, be handled by the analysis tool (server status handler) on the load balancer, which has one file to allow it to correctly calculate the users spanning onto all real servers and the current available resources of real servers.

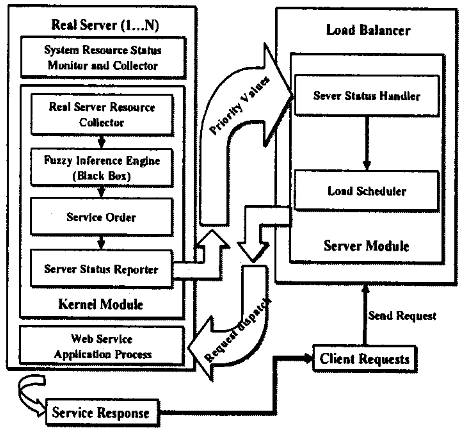

Figure 1 The Proposed Load-Balancing Algorithm

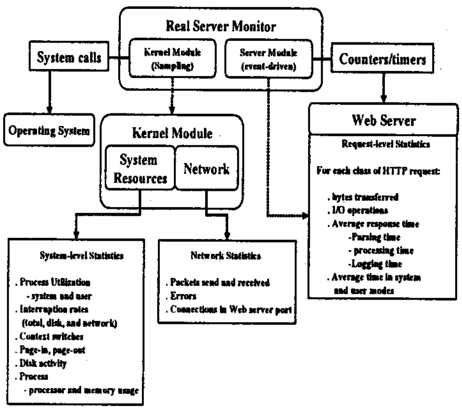

Figure 2 Overviews System Parameters of a Real Server

Figure 1 illustrates the basic architecture of the proposed load-balancing algorithm. Figure 2 shows the multiple system performance metrics of a real server [6]. These are monitored by RTSM (Real-Time Server Monitor) that developed in this work. It assigns different weights to real servers according to the states of their current available resources (H/W and S/W). It, then, estimates the whole workloads of real servers by putting a FIS between system resources and throughput. Since this scheme also considers the uncertainty of the network, the performance of the proposed load-balancing algorithm is more effective to the conventional ones. In the proposed algorithm, a real server determines its service order by itself using automatic report system of the load balancer. Thereby, we can minimize the overhead incurred by the frequent information transactions between the load balancer and real servers.

Real servers can also monitor their resources to send that the information to a FIS (Fuzzy Inference System) module obtained from ANFIS model. The FIS module estimates the service order based on the briefly trimmed information about the resources, and then immediately determines the orders. The reporter also transfers the results to the load balancer that has an additional module called the status handler of real servers, which manages the traffic conditions of real servers. The load balancer efficiently distributes the traffics throughout the load scheduler built on our proposed algorithm.

III. PERFORMANCE METRICS OF REAL SEVER

1. Experimental setup

In order to determine whether the servers can meet the increasing demands, a tool that can evaluate Web server performance is needed. A Web Benchmark is such a tool which can compare the performance of different web servers. It usually consists of a workload generator and a monitor. Web traffic is generated by the workload generator and the server performance is measured by WebBench monitor. We used WebBench 4.0 for generating the workloads. However, we not used the monitored results of WebBench. We use the monitored results on a real server that developed in this paper. This monitor is the RTSM (Real-Time Server Monitor). Figure 3 show our experimental setup for web server performance monitor. To run WebBench, this needs a minimum of three machines that are networked together [7]:

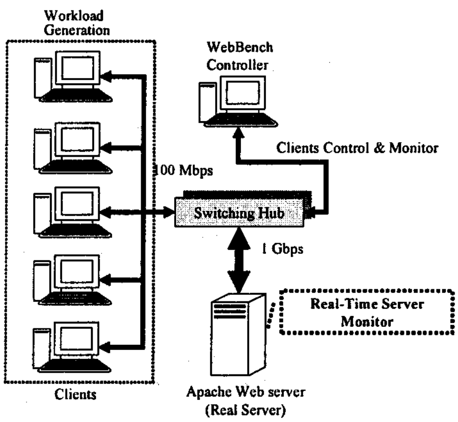

Figure 3 Experimental Setup for real server monitor

- *A Web Server*: This machine is real server that runs the Web server software (Apache Web server version 1.3.2).
- *A Controller*: This is a PC running Windows NT, Windows XP, or Windows 2000 and the WebBench controller program. It set up, start, monitor, and stop the WebBench tests from this PC. When the tests end, the clients send their results to the controller. The controller, unlike the clients, does not affect the server's overall score.
- *Clients*: These are the actual workloads generator. These are the PCs running the WebBench client program. WebBench supports Windows 95/98, Windows NT, Windows XP, or Windows 2000 clients. The clients execute the WebBench tests and send requests to the server.

2. Performance Metrics

Apache Web server is a general-purpose webserver, designed to provide a balance of flexibility, portability, and performance. Although it has not been designed specifically to set benchmark records, Apache Web server is capable of high performance in many real-world situations [8]. We used the default values of the Apache performacne parameters.

We have been monitored many facts of performance associated with real servers. Performance metrics are critical because they gauge the limit of a real server. We measured in four key aspects for the performance metrics of real servers; CPU overhead, the amount of memory usage, system load average and system throughput.

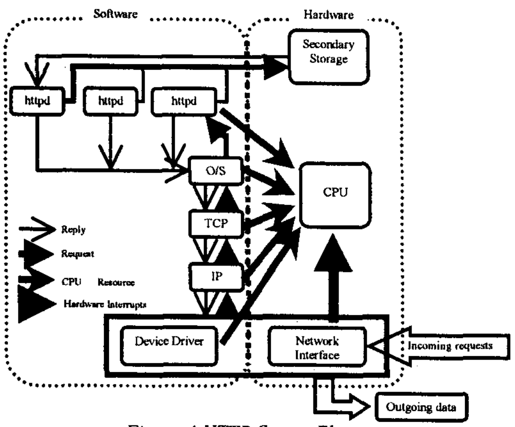

Figure 4 HTTP Server Elements

Figure 5 shows CPU overhead or utilization of a real server. All real servers require system resources to examine the incoming packets and to parse the HTTP requests, and it imposes overhead on network performance. As presented in figure 4, CPU has many tasks for each network connections, TCP/IP stacks and HTTP requests [4-5]. Therefore, CPU overhead reflects the most system state. Thus, CPU overhead is the most important performance metric.

Figure 6 show the amount of used memory state of system, when a real server provides services (homepage, FTP, streaming and etc.,) for the client's requests. This represents the total amount of used memory. The worst case in performance problem of real servers is the situation that do not swapping at disk for a whole server processes, when the amount of the available memory on real servers is lacking. If real servers get into this situation, the performance of those will drop significantly [5]. Therefore, the amount of memory usage is one of the important factors for the performance metric.

Figure 7 shows the load average of a real server. The load average is associated with the number of data requesting connections per second. Also, it is associated with the total number of running, sleeping, waiting network processes on real servers. Figure 7 represents the load average that is monitored when a real server provide the services for client's requests. It is important for the performance metric.

Figure 8 shows a real server throughput. Throughput is also an important metric associated with network process. Typically measured in bits per second, the throughput rates of the real servers are able to pass traffic through its internal network environment. While throughput is measured in bits per second, it is actually a combination of two different variables: packet size and packets per second. The number of packets per second is really the most important factor of a load balancer or any network devices. The combination of it and packet size determines the bits per second.

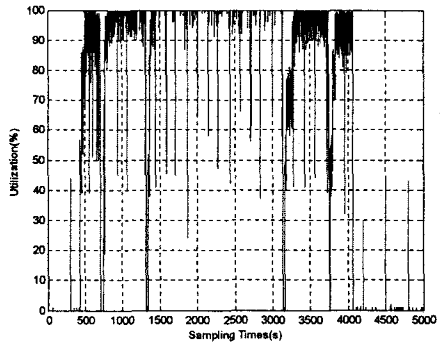

Figure 5 CPU Overhead of A Real Server

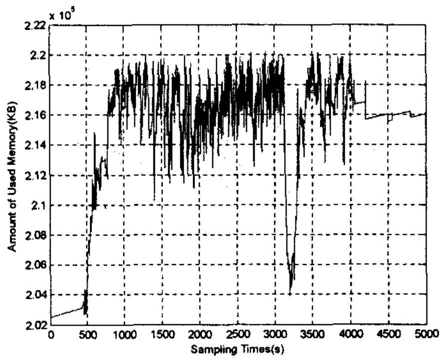

Figure 6 Amount of Memory Usage

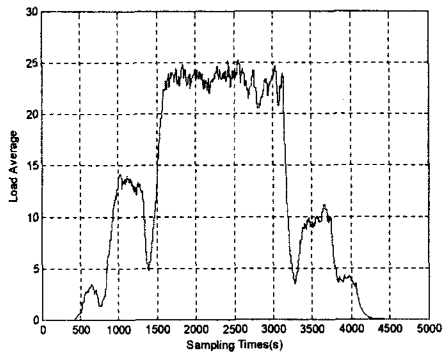

Figure 7 System Load Average

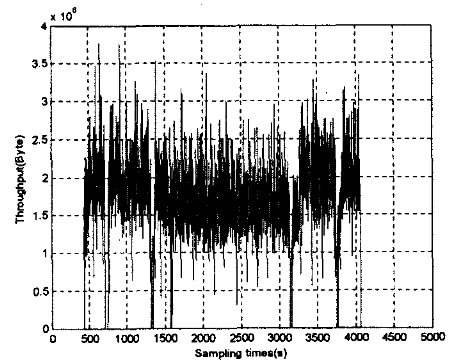

Figure 8 A Real Server Network Throughput

IV. MDELING AND SIMULATION RESULTS

In this section, we present experimental results in using the proposed algorithm.

Figure 9 shows the ANFIS modeling result. The inputs are the CPU overhead, the amount of used memory and the load average. The output is the network throughput (bit per second) including the total number of packet process and transmitted packet size of a real server. In our approach, requests are distributed to a real server by using the fuzzy load-balancing algorithm equipped with the DSR(Direct Server Return) routing algorithm that has a most excellent performance compared to the other routing algorithm(NAT, IP tunneling and etc.,).

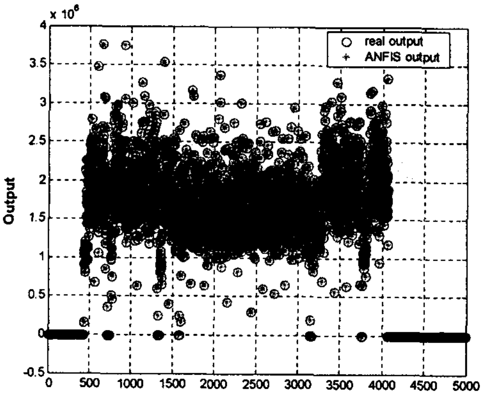

Figure 9 ANFIS Output (*) and Real Throughput (o)

The performance of the proposed load-balancing algorithm is reported in Figure 10. When each of clients requests an identical amount of network traffic the same as the static page (html) and dynamic page (CGI) requests, the proposed method in this paper illustrate the better than the conventional method

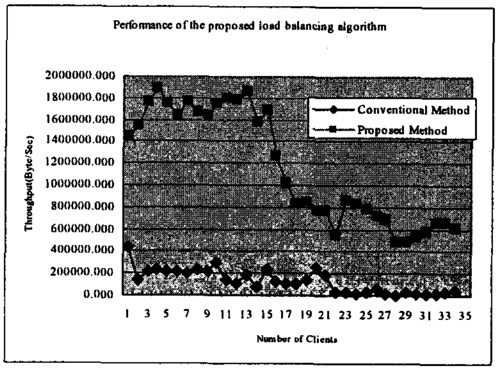

Figure 10 Performance of the Proposed Algorithm

V. CONCLUSION

The intelligent load-balancing is the term applied to load balancing approaches that make correct traffic distribution decisions based on parameters such as a real server status. In this way, requests are not redirected to a server that is not operational or unable to handle additional client requests. The intelligent load-balancing is more effectively accomplished on the use of a dedicated hardware or software load balancing solution.

The redirection of client's request will deliver the best user experience provides the benefit of performance. In the same manner, the local load-balancing redirection of client requests to a real server will make better performance. The proposed load-balancing algorithm that redirects incoming requests to a real server will provide the customer the best response time.

VI. REFERENCES

[1] A. Kumar, M. Singhal, M. Liu, "A model for distributed *decision making: An expert system for load balancing in* distributed system," *COMPSAC*, pp. 507-513, 1987

[2] P. Chulhye and J. Kuhl, "A Fuzzy-Based Distributed Load Balancing Algorithm for Large Distributed System" *Proc. 2nd Int'l Sym. Autonomous Decentralised Systems*, pp. 266-273, Apr. 1995

[3] Chin Wen Cheong, V. Ramachandran, "Genetic Based Web Cluster Dynamic Load Balancing in Fuzzy Environment", 0-7695-0589-2/00 2000 IEEE

[4] Ralf S., "Load Balancing Your Web Site Practical Approaches for Distributing HTTP Traffic", *http://www.webtechniques.com/archives/archives/1998/05/engelschall/*

[5] Xiaodong Zhang, "Improving Distributed Workload Performance by Sharing Both CPU and Memory Resources", *Department of Computer Science College of William and Mary Williamsburg, VA 23187-8795*

[6] Jissara A;meida "Measuring the Behavior of a World-Wide Web Server", *Technical Report CS 96-025*, October. 29, 1996

[7] WebBench On-Line Documentation and WebBench FAQ, *http://www.etestinglabs.com/benchmarks/webbench/webbench.asp?visitor=*

[8] Apache HTTP Server Version 2.1 Documentation, "*http://httpd.apache.org/docs-2.1/en/*"