A Web Load Distribution System: Implementation and Evaluation with PlanetLab

Cheng Tian* , Ryota Ayaki* , Hideki Shimada* , and Kenya Sato*

(Received February 5, 2013)

The Internet increases its popularity and evolves largely. The lack of a central coordination is a critically important problem to the rapid growth and evolution of the Internet. The lack of management makes it very difficult to guarantee the web servers keep proper performance and to deal systematically with performance problems. Meanwhile, the available network bandwidth and server capacity continue to be overwhelmed by the accelerating growth of the Internet utilization. When a web server is under a heavy load, its performance suffers, and it may even crash, which caused customers feel largely inconvenient and unsatisfactory. To avoid the web server performance problems caused by high load, a few methods and systems have been proposed. But, some of them are commercial, and some of them cannot solve the problem well. We think that distributing its load to other servers that are not busy is a solution for the problem. In this research, we propose a web load distribution system made by a network technology that can distribute the web load from the web server to other servers automatically, when the web server is in high loads. We implement the system on PlanetLab, which is a free global research network that supports the development of new network services. Then we evaluate the system in two cases. Based on the evaluation results, we have learned that the HTTP response delay was reduced. So we verified that this technology can distribute the web server load so that users can access their web servers faster during high load time.

Key words : Web system, load distribution, PlanetLab, CDN

1. Introduction

Due to the Internet's continued popularity, sometimes users experience slow server response. Their servers might even crash, so that it cannot provide any web services, which is very inconvenient. From the company side, commercial losses are even more substantial. To avoid such web server performance problems caused by high load, we set up shared cache servers called assistant servers all over the world in our system to distribute web server load. When a web server is experiencing a high load, user requests are automatically transferred to assistant servers who distribute the load and spread web server resources all over the world to access web servers that are faster abroad.

After Section 2, related works and existing systems are introduced. Our proposed system is presented in detail in Section 3. In Section 4, the implementation is explained, and the evaluation is shown in Section 5 and we also compare the proposed system with existing systems.

2. Related Works and Existing Systems

In networking, load balancing is a technique to distribute the server load evenly across two or more servers, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy. The load balancing service is usually provided by a dedicated program or hardware device, such as a DNS server. It is commonly used to mediate inter-

* Information and Computer Science, Graduate School of Science and Engineering, Doshisha University
1-3 Tatara Miyakodani, Kyotanabe-shi, Kyoto, 610-0321 Japan
Telephone:+81-774-65-6297, Fax:+81-774-65-6801, E-mail:ksato@mail.doshisha.ac.jp

nal communications in computer clusters, especially high-availability clusters. If the load is more on a server, then the secondary server takes some load while the other is still processing requests.

2.1 *Related Works*

Load balancing is a critical issue for distributing web server loads. One of the thrust of such research spreads the entire contents to network nodes more evenly, and can find out the parts of the contents more quickly when the user wants to get them. In peer-to-peer (P2P) networks, the distributed hash table (DHT) has been researched for load balancing. Chord[1], Pastry[2], Tapestry[3], and Content-Addressable Networks (CAN)[4], which we call DHTs, are the most famous algorithms that support DHT functionality. In such structured systems, a unique identifier is associated with each data item and each node in the system. The identifier space is partitioned among the nodes that form the P2P system, and each node is responsible for storing all the items that are mapped to an identifier in its portion of the space. Thus, the system provides an interface consisting of two functions: put(id, item), which stores an item, associating with it a given identifier id; and get(id), which retrieves the item with the identifier id[6].

2.2 *Existing Systems*

To reduce the web server load, a few systems have been proposed to distribute the web load to other servers. Proxy server[5], Round-robin DNS system[7], load balancer[8], content distribution network (CDN)[9] , are the most popular examples.

2.2.1 Proxy Server

In computer networks, a proxy server acts as an intermediary for requests from users seeking resources from other servers and copies these resources to respond the requests faster to users. But proxy servers also have problems. Their performance might suffer during high load times like a web server, because they are usually comprised of only a few machines.

2.2.2 Round-robin DNS

A round-robin DNS system is one of the load distribution and balancing techniques. In its implementation, it responds to DNS requests not with a single IP address but with a list of IP addresses of multiple servers. But it responds to the IP addresses from a list's fixed order and needs extra machines for distribution because copies of the web server contents are stored on these machines in advance.

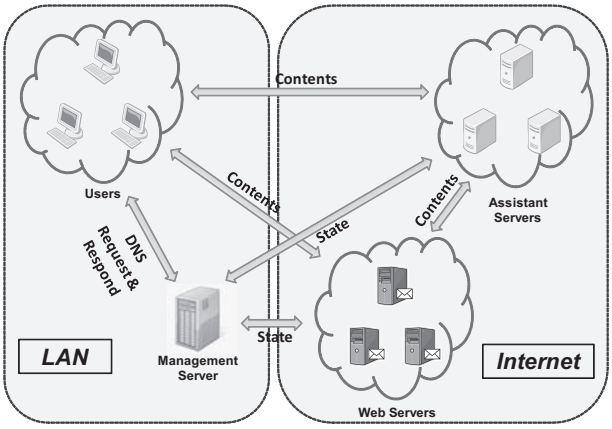

Fig. 1. System Architecture.

2.2.3 Load Balancer

A load balancer is working like a management server for load balancing. It manages the requests from the external network, and forwards these requests to multiple servers with equivalent functionality. The load balancer sends distribute these request to multiple servers as many as possible for maintaining a high speed of each server response. Also, if users need to communicate with the server multiple times by one set of transactions, the load balancer will also need an ability to transfer communications from the same server as the client is always the same. Generally, a server can be used as a load balancer by introducing a dedicated application.

2.2.4 Content Delivery Network

A content delivery network is a system of computers containing copies of data, placed at various points in a network so as to maximize bandwidth for access to the data from clients throughout the network. A client accesses a copy of the data near to the client, as opposed to all clients accessing the same central server, so as to avoid bottlenecks near that server. One benefit of it is that the capacity sum of strategically placed servers can be higher than the network backbone capacity. For instance, when there is a 10 Gbit/s network backbone and 100 Gbit/s cen-

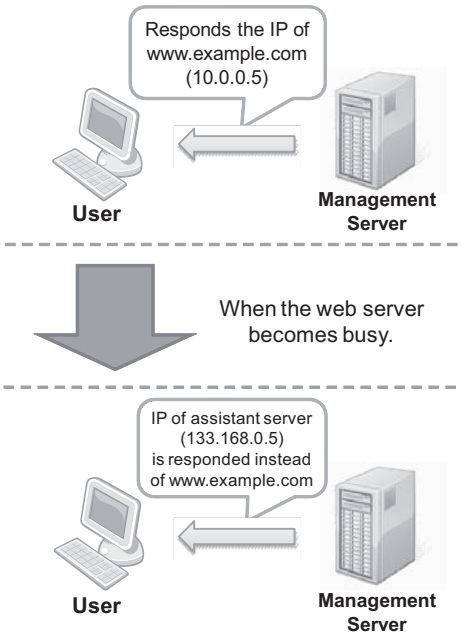

Fig. 2. IP address changed to the assistant server when the web server is busy.

tral server capacity, only 10 Gbit/s can be delivered. But when 10 servers are moved to 10 edge locations, total capacity can be 10 × 10 Gbit/s.

Akamai[10] is one of successful commercial CDNs. Its proprietary system, called Dynamic Site Delivery, is combined with high-speed network and cache servers. Based on the access request from a user, Dynamic Site Delivery can automatically determine the most appropriate server to provide the content faster. Technically, Akamai's DNS servers can get the information about the users' locations basing on the IP addresses, and make users access the closest Akamai's servers instead of accessing to the servers which are in USA. The benefit is that users can receive content from whichever Akamai server is close to them or has a good connection, leading to faster download times and less vulnerability to network congestion or outages.

3. Web Load Distribution System

The purpose of our proposed web load distribution system is to reduce the web server load and automatically distribute it to other assistant servers to maintain balanced web access. When it is busy, it also makes users directly access the assistant servers instead of the web server. For managing these machines and accesses and distributing the load, this system features a management server and assistant servers. In this section, we will describe the proposed system in detail. Figure 1 shows the system architecture.

3.1 *Management Server*

The management server, which manages the users, the web servers, and the assistant servers, has three main functions. First, it receives requests from users and the state data from the web server. Second, based on the state of the web server, the management server judges whether it should provide a new assistant server to the web server to distribute the load. There are two cases about the web server. Third, if the web server is not in a high load, the management server will do nothing. But, if the web server is in a high load, the management server will respond the user's DNS request with an IP address from the list of assistant servers (Figure 2). This way make the user does not access the web server, but directly accesses the assistant server when the web server is in a high load. Because the assistant server is not overload, it can respond the requests from users quickly.

For achieving these functions, there is a DNS server working in the management server. That makes the management server can respond the user's DNS request. A management program, which is working on the management server, can add an IP address of assistant server into the DNS zone file. And the proposed system can also support multiple web servers. If another web server need to use the proposed system, we just need to run management program once again, which is very simple. At the moment, there two management programs are running on the management server. Also, the state file names from the web servers are different to each other. Based on the filenames, the management server can determine the state is belong to which web server. We will explain it in some detail in the Section 4: Implementation.

3.2 *Web Server*

Web servers in the proposed system, working like common web servers, provide web services to

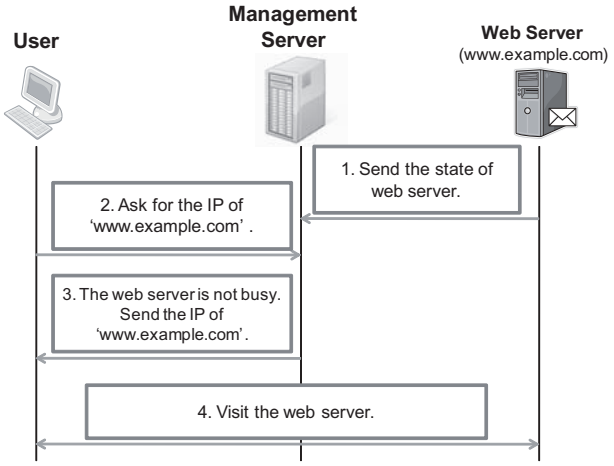

Fig. 3. Action when the web server is not busy.

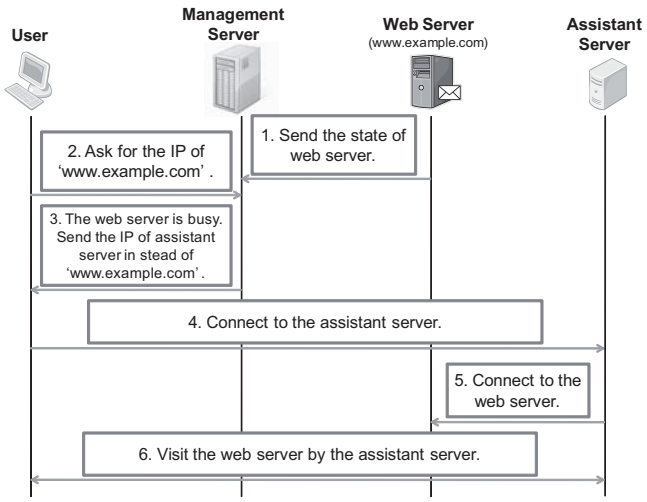

Fig. 4. Action when the web server has high load.

users. But there is a difference between the common one that a state checking and sending program is running on the web server in the proposed system. Our web servers are working by the web service software. The program can check the current quantity of established connections from the web service software and save it into a file. Then the number is sent to the management server by sending program. Certainly, we make the state file name be different to identify different web serers.

3.3 *Assistant Server*

Assistant servers are working like a cache system about web server for responding fast. On the other hand, sometimes when we access the oversea web server, we feel the response is very slow. Maybe the high load cause the response delay at that moment, but we also think the long distance and complicated routing between the user and the web server perhaps is the main cause. To improve it, we made assistant servers distributed around the world. Because of that, the user can access the closest assistant server without accessing oversea. An assistant server's main function is getting the contents from the web server and sharing them with users to reduce the web server load. Also, assistant servers are sending their state file like the web server to the management server to ensure that their are not in a high load. If one assistant server is overloaded, the management server will change it into another idle assistant server.

3.4 *Action of the Proposed System*

In this subsection, we will explain the action of the proposed system. Figures 3 and 4 show the temporal analysis about two cases of system main action, witch are the web server is or is not overloaded.

The following is the action when the web server is not overloaded. Figure 3 shows the temporal analysis about it.

1. In the proposed system, the web server always sends a file about its condition to the management server.
2. A user asks for the IP address of 'www.example.com' when he want to visit it.
3. The management server receives the user request and checks the web server state. Because the web server is not busy at the moment, the management server sends the web server's IP address.
4. The user gets the web server's IP address by DNS respond from the management server. Then he can access the web service of 'www.example.com'.

The following action occurs when the web server is overload. Figure 4 shows the temporal analysis about it. Steps 1 and 2 are identical as when the web server does not have a high load. We explain it from the step 3.

Fig. 5. Distribution about the Notes in PlanetLab.

3. Based on the user's request, the management server checks the web server condition. At the moment, since the web server has a high load, the management server selects one of the IP addresses of the assistant server instead of the web server and sends it as DNS response to the user.

4. The user gets the IP address of the assistant server and uses it to visit 'www.example.com'. However, the user does not know he is visiting the assistant server instead of the web server.

5. Based on the user request, the assistant server searches its own database to find the 'www.example.com' cache at first. If there is no caches matching the request, the assistant server connects to the web server and gets these contents from it.

6. The assistant server responds and sends these contents to the user. At the same time, it copies thees contents into its database for responding more faster after.

4. Implementation

4.1 *Implementation Platform: PlanetLab*

As the implementation platform of our proposed system, we chose PlanetLab[11], which is a global research network that supports the development of new network services. There are a lot of note that can be used around the world. In the proposed system, the implementation of assistant servers requires computers set up around the world. Using

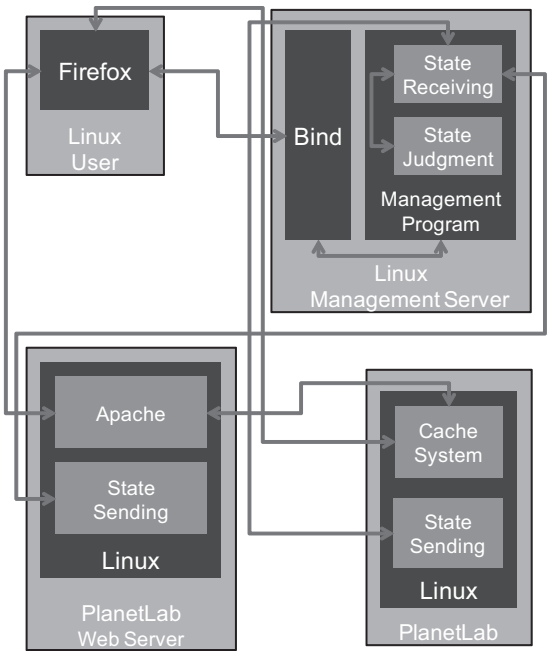

Fig. 6. Implementation Method.

PlanetLab, we can easily operate many computers around the world by such SSH applications; this would probably be impossible in a single laboratory. Almost all of PlanetLab's notes use Linux as their operating system. Therefore, we can use much developing software to create our system much more easily due to its great compatibility. Figure 5 shows the distribution about the notes in PlanetLab.

4.2 *Implementation Method*

Figure 6 shows the implementation method of the system.

We chose Firefox as our Internet browser for our test. Using Firebug, a development tool of Firefox, we determined the HTTP response time from the web server to users. Also, we set the management server's IP address to the IP address of the user's DNS server. Therefore, the user could send DNS requests to the management server to get the web server's IP address that he wanted to access.

The management server is built on Linux, can receive the web server and assistant servers' state by the state receiving program. The state judgment program can judge whether they are busy or not. Based on the result, the management program selects an appropriate IP address between the web and

assistant servers. Then it reports the IP address to the DNS software Bind, and the Bind will respond to the user with the IP address.

The web server is built on a PlanetLab node that responds to the user's HTTP request by web server software, Apache 2. The State Sending program are sending the state of web server to the management server.

The assistant server, which is built on a PlanetLab node, too. It is also sending its state to the management server. By doing that, when an assistant server becomes busy, the management server can know that, then changes the user's requests to another assistant server. A cache system, likes a reverse proxy system, is working on it to get data from the web server based on user requests. The cache system backups these resources for faster requesting when it sends them to users. Therefore, the assistant server can directly send the data to other users if the data have been cached before.

4.3 *The Programming*

The proposed system was built in Linux, and we programed the system by program language C. The state sending program, running on web servers and assistant servers, and the state receiving program used the socket to send and receive the date with each other. They are sending the state every 10 seconds. The state judgment programs, which are running on the the management server, can learn the state of them and judge these servers whether they are busy or not. If the result is not busy, the system will do nothing. But, if the result is busy, the management program gets an IP address from a list of the IP addresses of assistant servers and puts the IP address into the DNS file which is the lists of the IP addresses of the web server. Thus, the management server can respond to the DNS request from the user with the assistant server's IP address when the web server is in a high load.

5. Evaluation

For evaluating the system, users sent HTTP requests to the web server to get static html files and gathered the HTTP response times by Apache JMeter[12]. We increased the number of HTTP requests from 1 to 400 per second so place the web server in a high load. Finally, we gathered and analyzed the HTTP response time data.

Table 1. Evaluation Environment.

	CPU	Memory	OS	Application	PlanetLab node
User	Atom 1.6 GHz	1GB	Ubuntu 8.04	Java 1.60_05 Apache JMeter 2.3.4 Firefox 3.5	No
Management Server	Core2 2.0 GHz	3GB	Fedora 8	Bind 9.6.1 gcc 4.1.2	No
Web Server			Fedora	Apache 2 gcc 4.1.2	Yes
Assistant Server			Fedora	Squid 2.7	Yes

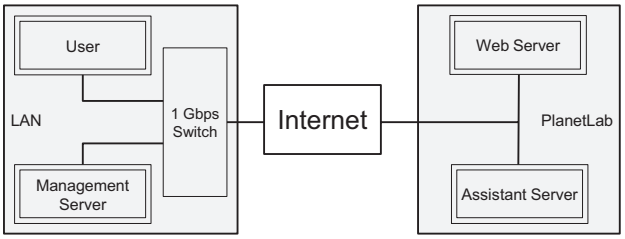

Fig. 7. Network Diagram.

5.1 *Condition*

Table 1 and Figure 7 show the evaluation environment and the network diagram of the system. Since the performance of the PlanetLab nodes was constantly changed because they are shared with many researchers, their condition is not presented. Instead, we recorded the evaluation times.

5.2 *Evaluation Results*

For evaluating the system, users sent HTTP requests to the web server to get static HTML files and gathered the HTTP response times by Apache JMeter[12]. We increased the number of HTTP requests from 1 to 400 per second so place the web server in a high load. Finally, we gathered and analyzed the HTTP response time data. The proposed system was evaluated in two cases, which the proposed server is working or not.

First, we made the proposed system be not running. The result of it is shown in Figure 8. The

scale of the vertical axis is the HTTP response time that the user received HTTP responses from the web server. The scale of the horizontal axis is the amount of requests which had been sent to the web server in one second. The HTTP response time from the web server rose by several seconds to 300 per second when the requests were sent by users. The web server performance decreased when it received 300 requests per second.

For faster access, we made our system be running. The management server judged the web server be in a high load and changed so that users got their contents from the assistant server. For a simple comparison, the management server judged whether the web server was in a high load when it received 400 HTTP requests per second. Based on the evaluation results Figure 9, we learned the following:

- The web server was stable until it received 200 HTTP requests per second.
- It became unstable when it received 300 HTTP requests per second.
- For simple comparison, the web server was judged to be in a high condition when it received 400 HTTP requests per second.
- Comparing with that before the management server connected the users to the assistant server, the HTTP response time delay was reduced .

  We compare the average HTTP response time in three cases, which are the web server is idle, overload NOT using the proposed system and overload using the proposed system. As a results, users get HTTP responses using the proposed system faster than without using it when the web server is overload.

5.3 *Comparison with Existing Systems*

Based on our evaluation, we have learned that the proposed system, working like a load balancing system, can distribute the web server load and reduce the HTTP response time when the web server has a high load. Unlike DHTs, our proposed system does not evenly distribute the entire contents to each node, but as far as possible, one assistant server saves the contents which are accessed by users to increase the hint percentage for requesting users more quickly. Also, there is a great difference between the proposed and existing systems.

- The proxy server cannot maintain web services for users when it crashes. Different to it, our system continues to provide the web service normally when one or more assistant servers crash, since it has more than one assistant server. Also using the proxy server, if we do not know the IP address of it, we cannot even access the web service and the Internet. But using the proposed system, we do not need to know the IP address of these assistant servers, because the management server responds the IP address to the user as DNS response.

- On the other hand, the round-robin DNS system requires setting up some extra machines, which are to keep the same contents as the web server in advance. The proposed system makes the assistant servers automatically get the contents from the web server based on the user requests. So different from the round-robin DNS system, the proposed system does not require any extra machines, which have the equivalent functionality as the web server in advance.

- Different to load balancer, when the web server does not have a high load, users can directly access the web server. But when the web server has a high load, users are made to access the assistant server by management server for faster access and distribution of the web server load. Also, the proposed system does not require any machines with the equivalent functionality as the web server in advance.

- Akamai, a CDN, is working by its Dynamic Site Delivery system. The proposed system is like it in general. But there also are some difference between them. Using the Dynamic Site Delivery system, that requires many servers which which are to keep the same contents as the web server in advance for requesting faster like the Round-robin DNS. The expenses will be very high for

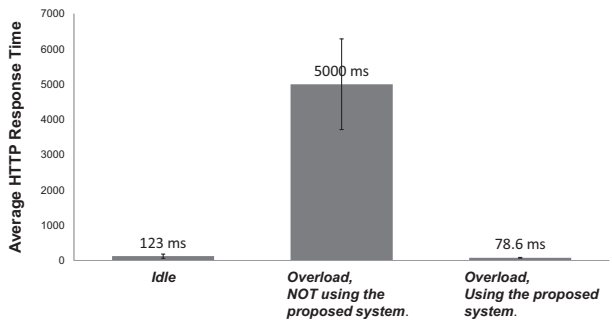

Fig. 10. Average HTTP Response Time in Three Cases.

setting up these extra servers and maintaining them. Using the proposed system, when the web server is overload, it requires some servers for distributing the load like the Dynamic Site Delivery system. But it will not require any extra server if the web server is idle. Certainly, the scale and performance of Dynamic Site Delivery system is much high than the proposed system. But, we think the proposed system fits these web servers those requires distributing the load but do not want to pay a large expenses.

6. Conclusions

In this paper, we proposed a web load distribution system that could distribute the web server load to provide faster access to users during high load periods. We also presented its implementation and evaluation. Our proposed system allows users to access an assistant server for faster access when the web server has a high load. Based on user requests, the assistant server gets the contents from the web server and copies them for faster response before sending them to users. Therefore, users get their web server contents faster. The contents are also distributed to other servers to ease the web server load. With this system, users will not experience any difference, even when the web server is busy. Based on implementation and evaluation, users get HTTP responses using the proposed system faster than without using it (Figure 10). We are thinking the proposed system can distribute the web server load to other servers when the web server is in a high load. So using the proposed system, however high the web server load is, the user can access the web contents smoothly.

References

1) I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," Proc. ACM SIGCOMM'01, 149–160, (August 2001).

2) A. Rowstron, and A. Rowstron, "Pastry: Scalable, distributed object location and routing for large-scale peer-topeer systems," Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms, 329–350, (November 2001).

3) B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, (April 2001).

4) S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," Proc. ACM SIGCOMM, 161–172, (August 2001).

5) Y. M. Xu, Z. S. Xiao, and S. Liang, "Study and application about technology of web cache server," Proc. Computer Engineering and Design in Shandong University, Vol.26, No.1, 126–128, (January 2005).

6) B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in Dynamic Structured P2P Systems," Proc. IEEE INFOCOMM 2004, Vol.4, 2253–2262, (March 2004).

7) V. Cardellini, M. Colajanni, and P. S. Yu, "Dynamic Load Balancing on Web-server Systems," IEEE Internet Computing, Vol.3, No.3, 28–39, (June 1999).

8) C. B. Coughlin, and S. Diego, "Network Load Balancing," United States Patent Application Publication, No.US 2004/0024861 A1, (February 2004).

9) G. Peng, "CDN: Content Distribution Network," Technical Report TR-125 of Experimental Computer Systems Lab in Stony Brook University, (February 2008).

10) E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: A Platform for High-Performance Internet Applications," ACM SIGOPS Operating Systems Review, Vol.44, Issue 3, 2-19, (July 2010).

11) PlanetLab, An opne platform for developing, deploying, and accessing planetary-scale services, `http://www.planet-lab.org/`.

12) D. Nevedrov, "Using JMeter to Performance Test Web Services," Published on dev2dev (`http://dev2dev.bea.com/`), (August 2006).

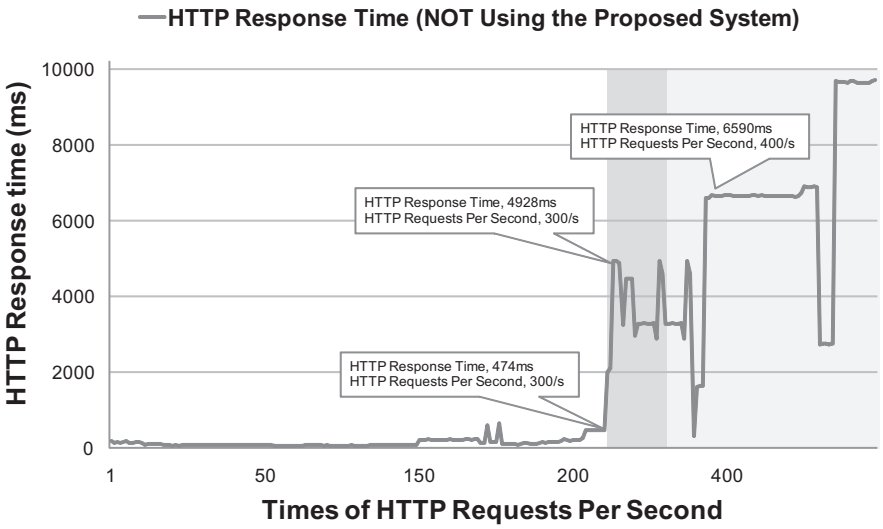

Fig. 8. Evaluation Results (NOT Using the Proposed System).

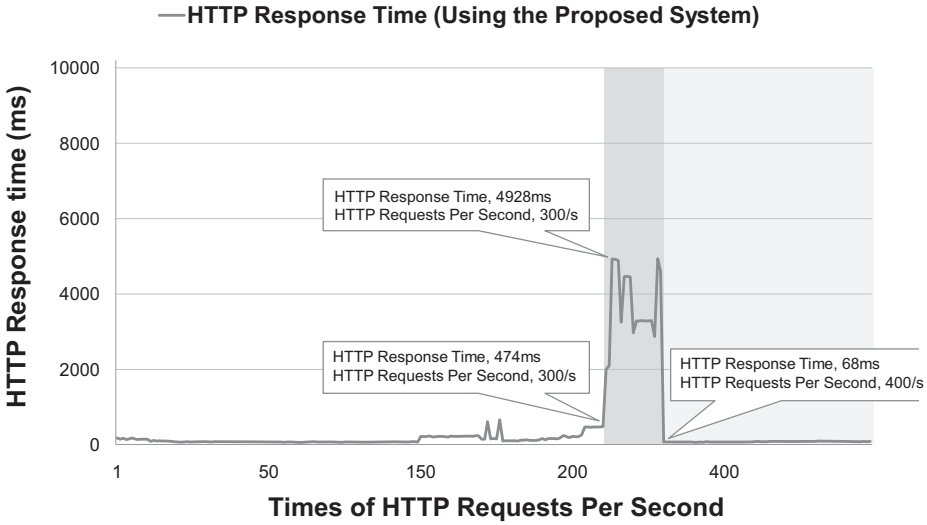

Fig. 9. Evaluation Results (Using the Proposed System).