Improvement of Load Balancing Method in a Distributed Web System Using DNS

Kota MORIGAKI and Keizo SAISHO
Kagawa University
2217-20 Hayashi-cho, Takamatsu 761-0396, Japan
s16g473@stu.kagawa-u.ac.jp, sai@eng.kagawa-u.ac.jp

ABSTRACT

Progress of virtualization technology in recent years made it easy to build virtual servers on cloud. They can be used as cache server for load balancing. However, expected responsiveness cannot be gained with insufficient cache servers against load. In contrast, costs will increase by surplus cache servers against load. Therefore, we have been developing a distributed web system that adjust the number of cache servers according to load of them to reduce running cost. In this study, a load balancing method using DNS round-robin is now developed. However, load imbalance occurs among the servers with this method and responsiveness decreases because it is difficult to distribute the load uniformly using DNS round-robin. Therefore, we implement a function to suspend the allocation of requests to the overloaded server. This paper describes improvement of load balancing method and evaluation of it. From results of experiments, we confirm that improved function is possible to prevent lowering responsiveness with lower TTL value.

KEYWORDS

DNS Round-Robin, Distributed Web System, Load Balancing, Suspend Allocation, Auto-scaling, Cache Server

1 INTRODUCTION

In recent years, the Internet users increase and much service is performed using Web. Therefore, load of Web servers is growing more and more. If the load is over the limit of server’s capacity, it returns the response with large delay, and it goes down in the worst case. Load balancing techniques are often used to avoid overload of servers. There is a Web system that distributes requests to multiple servers such as cache servers or mirror servers for load balancing. Progress of virtualization technology made it easy to build virtual servers on cloud. They can be used as cache server. Responsiveness, however, does not improve with insufficient cache servers against load. In contrast, costs will increase with surplus cache servers against load. Therefore, we developed a distributed web system using Load Balancer that dynamically adjust the number of cache servers according to load of them to reduce running cost[1]. In this study, we develop a method to distribute load to cache servers using DNS round-robin. However, load imbalance occurs among the servers with this method and responsiveness decreases because it is difficult to distribute the load uniformly using DNS round-robin. Therefore, we implement a function to suspend the allocation of requests to the overloaded server.

2 RELATED WORKS

A cloud auto-scaling mechanism aiming at providing necessary resources at low cost has been studied[2][3]. In [2], auto-scaling mechanism based on workload information and performance desire is implemented in Windows Azure platform. The result of the experiment shows that cost can be reduced by choosing an instance type of appropriate performance for the workload. This research covers a variety of applications, but our system targets only web application and aims at cross-use multiple cloud services.

In [3], an auto-scaling algorithm based on the number of active sessions of the web server is described. A load balancer is used for load balancing. Although we also use the number of active sessions as the load value, we use DNS for load balancing.

In [4], dynamic load balancing method using dynamic DNS update and round-robin mechanism is proposed. In this method, a server is dynamically added to or removed from the DNS list. The scheduling algorithm considers usage rates of server's CPU, memory, and network. The result of the experiment shows that both the response time and the average file transfer rate of the proposed system are faster than those of a pure round-robin DNS. It is similar to our load balancing method, but ours uses the number of active sessions as the load value.

3 DISTRIBUTED WEB SYSTEM USING DNS

Figure 1 shows our distributed Web system using DNS which consists of management server, authoritative name server, origin server and cache servers on cloud. The origin server services original contents and cache servers service the cache of them. The managing server manages the number of cache servers and DNS zone of the authoritative name server. For load balancing, this system uses DNS round-robin method which sends the list of IP addresses in a different order to a new client each time. Most clients use the first IP address they receive to connect server. Therefore, requests from clients are sent to each server. By managing the DNS zone of the authoritative name server, it is possible to control the start and stop of allocating request to each server.

The management server has the following functions.

- Load monitoring function
  The load monitoring functions monitors load of the origin server and cache servers. This function periodically measures the current and the maximum number of Web server processes and calculates ratio of the current number against the maximum number (Operating Ratio), and calculates average of Operating Ratio of working servers (Average Operating Ratio, AVGOR). This system uses AVGOR as load value.
- Cache server management function
  The cache server management function boots up and shuts down cache servers. This function decides the number of required cache servers based on AVGOR obtained by the load monitoring function. When AVGOR is greater than threshold of scale-out (Th_{high}), it boots up a new cache server. When AVGOR is less than threshold of scale-in (Th_{low}), it shuts down a latest booted cache server.
- DNS management function
  The DNS management function manages the DNS zone of authoritative name server. According to the booting up and shutting down cache server, IP address of it is added to or removed from the DNS zone dynamically. When the load monitoring function cannot monitor load of a server, it also removes the server. Therefore, it is possible to cope with server failure such as system down.

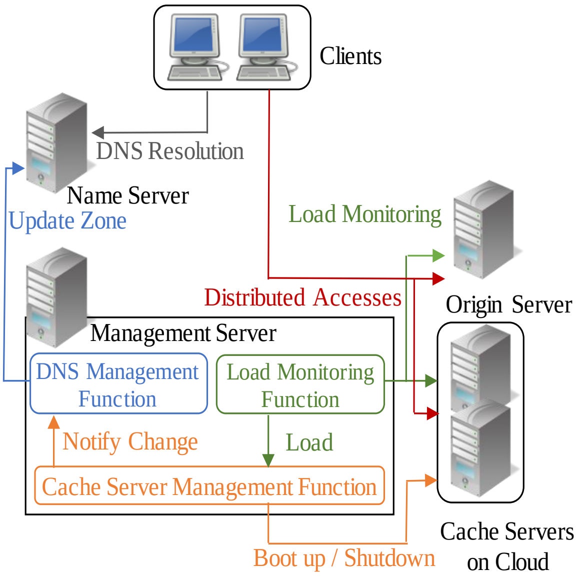

Figure 1. Distributed Web System using DNS

4 LOAD BALANCING USING DNS ROUND-ROBIN

We experimented with the distributed Web system described in the previous section. The result showed, load imbalance among the servers using the DNS round-robin method.

4.1 Experiment Environment

Figure 2 shows the experiment environment. All servers and clients are built as virtual machine on hypervisors which specifications are shown in Table 1. The management server and DNS servers are built on hypervisor1, the origin server and nine cache servers are built on hypervisor2 and hypervisor3, and twelve clients are built on hypervisor4. Mirror server is used instead of cache server because cache mechanism is now developing. Apache2.4[5] is used as a Web server software. DokuWiki[6] runs on all servers. Each client accesses the web server using Siege[7]. Siege is the stress test tool. The number of simultaneous accesses is set to 100. Therefore, the maximum number of simultaneous accesses is 1,200 (100×12). TTL value for DNS is set to 60 seconds. Th_{high} and Th_{low} are set to 0.6 and 0.1, respectively.

4.2 Experiment Procedure

The scenario of the experiment is shown in below. To examine the load and the response time of each server, the number of simultaneous accesses to web servers is stepwise changed.

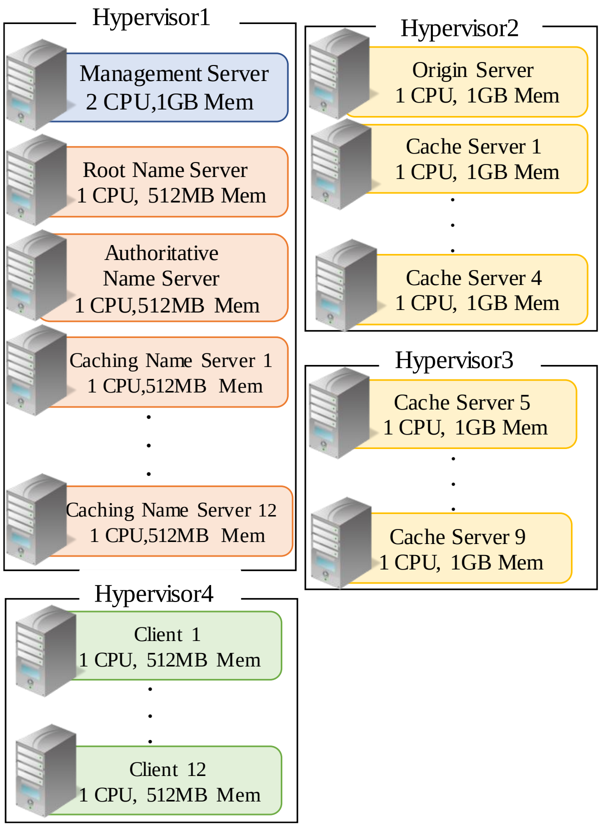

Figure 2. Experiment environment

Table 1. Spec of each hypervisor

	CPU	Memory
Hypervisor1	Intel Xeon E5-2620	32GB
Hypervisor2	Intel Xeon E5-2620	32GB
Hypervisor3	Intel Xeon E5-2620	32GB
Hypervisor4	Intel Core i7-4790K	32GB

I. Start with no accesses.
II. Add 1 client every 30 seconds.
III. After all clients are added, keep all clients accessing for 500 seconds.
IV. Remove 1 client every 30 seconds.
V. End when no accesses.

4.3 Experiment Result

Figure 3 shows Operating Raito of each server. Figure 4 shows the response time and the Operation Raito of the origin server.
In Figure 3, several servers are overloaded for a long time, and some servers remain low load. This phenomenon happens clearly around 500 seconds.
In Figure 4, purple line and blue line show average response time and maximum response time for one second, respectively, green line shows Operation Ratio. Maximum response time varies very much. Requests from clients are distributed to each server by using the round-robin method. However, this method cannot consider the load of each server and it causes load imbalance among the servers and response time lengthens.

5 SUSPENDING FUNCTION

We think that the problem described in the previous section can be coped with by suspending the allocation of requests to the overloaded server. We implement a function that excludes overloaded servers from DNS answer. We call this function suspending function. The function uses the PipeBackend of PowerDNS[8] that is DNS software. It can call external program that resolves DNS queries dynamically through PipeBackend module. We implement the program that resolves DNS queries based on the

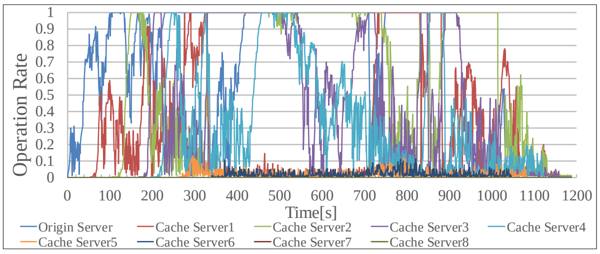

Figure 3. Operation Ratio of each server

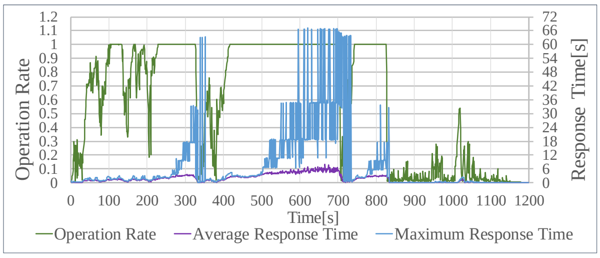

Figure 4. Operation Ratio and response time of origin server

configuration file shown in Figure 5. The file contains a hostname, IP address, status value of each server and TTL value. The status value is updated every second based on Operation Ratio of each server obtained by the management server and sent to the authoritative name server. The status value is set to 0 while the corresponding server stay in overloaded. otherwise, the status value is set to 1. The IP address is included in DNS answer if the corresponding status value is 1. In contrast, the IP address is excluded from DNS answer if the corresponding status value is 0. For example, IP address 192.168.11.21 and 192.168.11.22 is included in DNS answer with configuration shown in Figure 5.

6 EVALUATION

In this section, we evaluate the function described in the previous section. The experiment environment and the experiment procedure are the same as in Section 4. When the Operation Ratio of the server is the threshold value and over, the function decides that the server is overloaded and set the status value in configuration file to 0. Otherwise, the value is set to 1. In this experiment, the threshold value is set to 0.6 (case A), 0.8 (case B) or 1 (case C). Experiment without the suspending function is represented as case D. We performed experiments ten times in all cases. In order to investigate influence of the function to suspend allocation of requests to overloaded servers, the experiment results while number of simultaneous accesses is maximum are examined.

```
example.com:
  A:
    IP:
        192.168.11.21: 1
        192.168.11.22: 1
        192.168.11.23: 0
        192.168.11.24: 0
        192.168.11.25: 0
        192.168.11.26: 0
    TTL: 60
```

Figure 5. A Sample configuration file

The average of the results is shown Table 2. The cost is sum of uptime of all servers. The average response time in case D is the best in all cases. The number of requests per cost in case D is also the best.

Table 2. Experiment result of each case (TTL 60)

Case	Threshold value	Cost	Total response time	Total number of requests	Average response time	number of requests per cost
A	0.6	3454	445152	312404	1.42	90.45
B	0.8	3316	447405	309476	1.45	93.32
C	1	3279	454476	292067	1.56	89.08
D	Without function	3258	431490	304978	1.41	93.60

Table 3 shows the percentage of access with response time longer than or equal to 3,9,15,30 and 60 seconds. The blue and red letters indicate the lowest and highest percentage, respectively. Percentages of access with response time longer than equal to 3 and 60 seconds severally in case D are the highest. In contrast, other percentages in case D are the lowest.

Table 3. Percentage of long responded access (TTL 60)

Case	Threshold value	3	9	15	30	60
A	0.6	15.103	2.453	1.269	0.468	0.031
B	0.8	14.760	2.910	1.569	0.609	0.027
C	1	17.070	3.068	1.692	0.656	0.030
D	Without function	18.671	1.403	0.650	0.262	0.074

The results show the function is ineffective. In DNS, the TTL value specifies the expiration date of the DNS cache. It takes long time to reflect the updated DNS zone with large TTL.

Therefore, the suspending function takes no effect on response time. So, the same experiments except TTL value set to 30 seconds is performed.

The average of results is shown Table 4. The average response time in case C is the best in all cases. The number of requests per cost in case C is also the best.

Table 4. Experiment result of each case (TTL 30)

Case	Threshold value	Cost	Total response time	Total number of requests	Average response time	number of requests per cost
A	0.6	3745	417580	368184	1.13	98.33
B	0.8	3712	419933	364384	1.15	98.16
C	1	3627	416003	371359	1.12	102.40
D	Without function	3407	433705	320231	1.35	93.99

Table 5 shows the Percentage of long responded access. All percentages of access in case A are the lowest in all cases. The percentage of access with response time longer than equal to 3 seconds in case D is the worst in all cases and more than twice as high as that in case A.

Figure 6 and Figure 7 show the response time and the Operation Raito of the origin server in case A and case D, respectively. Line colors are same as in Figure 4. In Figure 6, the maximum response time is about half of the maximum response time in Figure 7. These results show the effectiveness of the suspending function with small TTL.

Table 5. Percentage of long responded access (TTL 30)

Case	Threshold value	3	9	15	30	60
A	0.6	8.567	0.817	0.400	0.103	0.0000
B	0.8	9.160	0.936	0.461	0.118	0.0000
C	1	10.268	1.009	0.442	0.122	0.0004
D	Without function	19.164	1.011	0.425	0.147	0.0360

7 CONCLUSION

We implemented the suspending function to exclude overloaded servers from DNS answer and evaluated it. By the experiment, it is confirmed that the function is possible to improve responsiveness with lower TTL value. However, the function reduces responsiveness with higher TTL value.

The followings are future works.

- Practical Scenarios of Experiment
- Examination of TTL value
- Experiment using cloud environment

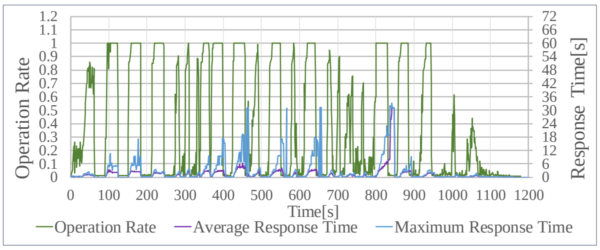

Figure 6. Operation Ratio and response time of origin server in case A (TTL 30)

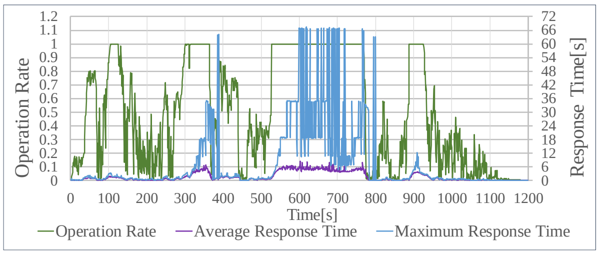

Figure 7. Operation Ratio and response time of origin server in case D (TTL 30)

REFERENCES

[1] A. Horiuchi, and K. Saisho. “Prototyping and Evaluation of Virtual Cache Server Management Function for Distributed Web System,” The 2015 International Conference on Computational Science and Computational Intelligence (CSCI'15), pp.324-329, 2015.

[2] M. Mao, J. Li, and M. Humphrey. “Cloud auto-scaling with deadline and budget constraints,” 11th ACM/IEEE International Conference on Grid Computing (Grid 2010), 2010.

[3] T.C. Chieu, A. Mohindra, A.A. Karve, and A. Segal. “Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment,” 2009 IEEE International Conference on e-Business Engineering, pp.281-286, 2009.

[4] JB. Moon, and MH. Kim, “Dynamic Load Balancing Method Based on DNS for Distributed Web Systems,” International Conference on Electronic Commerce and Web Technologies, pp. 238-247, 2005.

[5] Apache, https://httpd.apache.org/

[6] DokuWiki, https://www.dokuwiki.org/dokuwiki#

[7] Siege, https://www.joedog.org/siege-home/

[8] PowerDNS, https://www.powerdns.com/