Computer Science and Information Systems 12(2):765–786 DOI: 10.2298/CSIS141104025H

Dynamic Load Balancing Technology for Cloud-oriented CDN

Hui He[1], Yana Feng[2], Zhigang Li[2], Zhenguang Zhu[1],Weizhe Zhang[1], Albert Cheng[3]

[1] School of Computer Science and Technology,
150001 Harbin Institute of Technology, Harbin, HL, China
{hehui, wzzhang}@hit.edu.cn
[2] HLJ Province Electronic Information Products Supervision Inspection Institute,
150001 Harbin, HL, China
105104397@qq.com, lzg@nstc.gov.cn
[3] Department of Computer Science,
University of Houston
Houston, TX, USA
cheng@cs.uh.edu

Abstract. With soaring demands of Internet content services, content delivery network (CDN), one of the most effective content acceleration techniques, is applied into Internet services. Content routing functions in CDN are generally realized by load balancing system. Effectiveness of load balancing strategy determines response speed to users and user experience (UE) directly. This paper extracted the most important influencing factor of CDN loading from common network services and proposed the Variable Factor Weighted Least Connection. The proposed algorithm made real-time computing and dynamic regulation in considering of effect of network applications on server load index, performance changes of the server and workload changes. It has been applied in LVS kernel system successfully. The experiment confirmed that the CDN load schedule system with Variable-Factor-Weighted-Least-Connection could balance loads among cluster servers dynamically according to processing capacity changes of servers, thus enabling to provide users desired services and contents during high large-scale concurrence accesses of users.

Keywords: content delivery network (CDN), dynamic load balancing, LVS, Weighted Least Connection

1. Introduction

In 2014, number of websites in China has reached 2.73 million and Chinese netizen population has exceeded 632 million [1]. How to provide reliable services to such a big user population challenges existing Internet technology greatly. Satisfying user experiences depend on quick responses to desired services and contents of users during high large-scale concurrence accesses. With the continuous expansion of Internet scale, internet content providers become aware that it is more and more difficult to provide desired contents to users within short response time. Such difficulty further intensifies upon unexpected flash crowd.

The traditional web service acceleration strategy which only sets cache at server is no longer enough to provide content acceleration for new network applications, such as blog, microblog, SNS, etc. Moreover, cache content changes frequently and has relatively limited acceleration capacity. Therefore, content acceleration technique in heterogeneous network environment is attracting increasing research attentions. Research data reported that users' patience to website response shortens from 8s to 4s [2]. To provide quick content distribution and transmission in content agency services, content delivery network (CDN) is developed [3]. It adds an overlay network into existing networks and distributes website contents to the "network edge" where is close to users. When users visit the website, desired content will be transmitted to them quickly through some algorithms and technologies. This is known as proximity access. CDN could accelerate content transmission, alleviate backbone network congestion and shorten response time of the website. The cloud-oriented CDN divides communication between Internet users and Internet resource providers into two parts: 1) interaction between users and content proxy server in CDN; 2) interaction between content proxy server and source content server. Such division not only makes the source service providers oriented at service delivery to users and content delivery network, which could reduce operation costs, deployment difficulty and management complexity of source service providers significantly, but also creates a great CDN market.

Compared to other technologies, cloud-oriented CDN technology has several advantages:

1. Completely transparent visit to the website. No manual choose or configuration is needed.
2. Good redundancy mechanism: there are multipoint redundancies within the physical area and on the network, which ensures that failure of one node won't affect normal visit of users.
3. Simple deployment and content management. It could be deployed without any modification to the source station.
4. It could examine availability of each node and delete unavailable node in time, thus increasing the web usability.
5. Higher content delivery quality, speed and availability.
6. Lower infrastructure and management facility costs of the whole website.
7. Lower pressure and load on source server.
8. Higher safety of the website. It could hide actual source server better and resist to DDos attacks better.

CDN includes four key technologies, namely, content routing, content delivery, content storage and content management [4]. Content routing functions in CDN are generally realized by load balancing system. Studying dynamic load balancing technology in content access system could effectively reduce bandwidth consumption of intermediate network and visit timeout, and improve utilization of server resources and the overall service ability of the cloud content proxy server system. This is conducive to safety and reliability of the content service-oriented distributed system platform, enabling it to bring more benefits to both users and enterprises.

The following text includes four chapters. Chapter II introduces existing researches on load balancing. Chapter III describes the Variable-Factor-Weighted Least Connection. Chapter IV tests the proposed Variable-Factor-Weighted Least Connection

by four design plans. It is confirmed valid. Chapter V is conclusions and future prospects.

2. Related works

Load balancing technology is to assign loads of one task to multiple servers evenly when one server is inadequate to accomplish the task. Load balancing mainly involves three problems: load parameter selection, load computing method and load scheduling strategy [5].

Load balancing technology can be divided into dynamic load balancing and static load balancing according to load information used during allocation [6]. Static load balancing makes decisions according to priori knowledge and running status of equipments and programs. Dynamic load balancing adjusts task assignment according to collected load information instead of priori knowledge. Compared to static load balancing, dynamic load balancing could acquire load information of the system in time, balance loads better through the scheduling strategy and increase the overall service capacity of the system. This explains the better application effect of dynamic load balancing.

Load balancing technology also can be divided into hardware load balancing technology and software load balancing technology according to the implementation way. Software load balancing installs software on the existing operating system of server. Hardware load balancing adds one or some hardware except for the server and network equipments. Loads are distributed by the added hardware. Selection of hardware load balancing or software load balancing is a problem similar with system structure problems of computer. Some part could be realized by both software and hardware. Compared to hardware load balancing, software load balancing has lower expenses and better flexibility, but poorer efficiency. Hardware load balancing has poorer flexibility due to the limited scalability, but is superior in efficiency. Appropriate implementation way shall be selected according to costs, efficiency and expandability.

Load balancing technology can be divided into local load balancing and global load balancing according to sphere of influence. Local load balancing assigns tasks within a local scope (generally within the same physical extent). Global load balancing assigns tasks within a larger scope. Generally, it uses hierarchical task assignment. For example, the task is assigned to several regions firstly and in each region, it is further assigned according to local load balancing. Global load balancing can be divided into two types: (1) load balancing based on measurement, such as CDN load balancing technology based on distributed Binning strategy (Jia Bo et al.) [7-9] and CDN load balancing technology based on performance measurement (Zhang Guomin et al.) [10]; (2) load balancing based on domain and classification, such as CDN load balancing based on new multicast characteristics of IPV6 (Zhu Tiannan et al.) [11], level-three load balancing strategy for streaming media that centered at cooperative interaction of peer domains (Zhang Guomin et al.) [12].

Brighten Godfrey et al. solved load balancing problem of dynamic P2P network through heat and loads of log files [13]. Zhu Binjie et al. addressed dynamic load balancing problems in P2P-CDNs by improving the lookup algorithm [14].

Specifically, Onur Destanoğlu et al. proposed a dynamic load balancing algorithm based on hydrodynamic model [15]. Chen Yan et al. put forward a load balancing algorithm based on air pressure model [16]. Ralf Diekmann et al. developed the nearest neighbor scheduling algorithm [17]. Sagar Dhakal et al. proposed the improved One-Short algorithm that takes link delay into account [18].

Network implementation layers include IP layer, HTTP layer and intelligent DNS layer [19-21]. Implementation layer could be selected according to specific services.

3. Variable Factor Weighted Least Connection load scheduling

In LVS, the only dynamic load information that could be acquired is link numbers of real servers (recorded upon request of passing the load balancer and controlled by corresponding timeout mechanism). It is simple to operation and doesn't need to request load information from real servers. However, it also has a serious shortcoming. Link number could only represent load information of few real servers and couldn't reflect actual loads of all real servers.

For CDN in backbone network, the Weighted-Least-Connection could reduce workload of load balancer effectively. But existing Weighted-Least-Connection neglects effect of different network application connection on server load. Hence, this chapter improves the Weighted-Least-Connection from load metric selection and load compute mode. Relationship of load metrics was analyzed by an experiment.

3.1. Selection and measurement of load parameters

In computing cluster emphasized on scientific computing task, dynamic load balancing is to solve maldistribution of machine loading caused by different operation task queue lengths of different servers. It could acquire all operation task information, thus enabling to adjust operation task queue of servers continuously during running. As a result, it could accomplish all operations in the shortest possible time. However, dynamic load balancing in content service agency couldn't predict time and scale of network request. Meanwhile, one request that has been assigned to one server won't be readjusted to other servers in view of costs. Compared to dynamic load balancing emphasized on scientific computing, dynamic load balancing in content service agency couldn't predict future task size and has higher requirement on real-time performance.

Actually, loads in content agency reflect busyness of servers which could be described by resource utilization of servers. Therefore, resource utilization of servers in agent-based cloud by different network applications should be analyzed firstly to find out the best index of busyness of servers.

Currently, mainstream internet business includes two categories: 1) frequent interaction of small data size, such as search engine, microblog, mail service, instant messaging, and so on; 2) continuous interaction of big data size, such as video websites, P2P download, etc. The following text will make an experiment on these two kinds of network application to observe resource utilization and provide experimental basis for selecting load computing indexes.

Load analysis for frequent interaction of small data size. An experimental environment was created on Linux to measure actual resource utilization. Configurations of the implementation platform are shown in Table 1.

Table 1. Configurations of the experimental environment

Configuration Name	Content
Operating system	Centos 6.2
Memory size	32G
CPU	4 AMD Opteron(tm) Processor 6136 CPU
Network card	GB Ethernet card
Installing software	Apache+Mysql+PHP+Wrodpress

To observe resources utilization of servers, multiuser access was simulated by using the LoadRunner software of HP in the experiment [22]. LoadRunner simulated that 300 users click five blog pages without video in 5min. Information statistics, including link number, memory usage, CPU utilization, disk I/O as well as quantity and bytes of receiving and sending packages, were made every 5s. LoadRunner held each connection for 600s in the experiment in order to ensure connection continuity (It held connection even after finished information acquisition).

According to information statistics, the average total disk I/O of servers every 5s reads 7.3 when accessing to common pages. This indicates that common pages consume only a small amount of disk I/O resources. No independent graphic is presented in this paper.

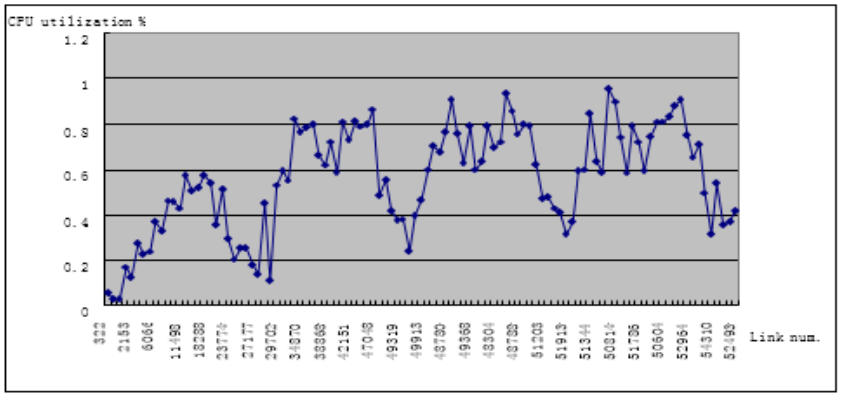

Fig. 1. Relationship between link numbers and CPU utilization when accessing to common pages

A proportional relationship between link numbers and CPU utilization is observed in Fig.1. This means that CPU utilization could reflect busyness of system (Although link number keeps increasing, most links are at TIME_WAIT after finished data acquisition.) Fig.2 shows that memory utilization also could reflect busyness of system. In Fig.3, the

total sending packages are far more than total receiving packages under high link number. It is important to note that in every 5s statistical time, total receiving packages basically conform to total sending packages (total sending packages is some times more than total receiving packages), but bytes of sending packages are significantly higher than those of receiving packages.

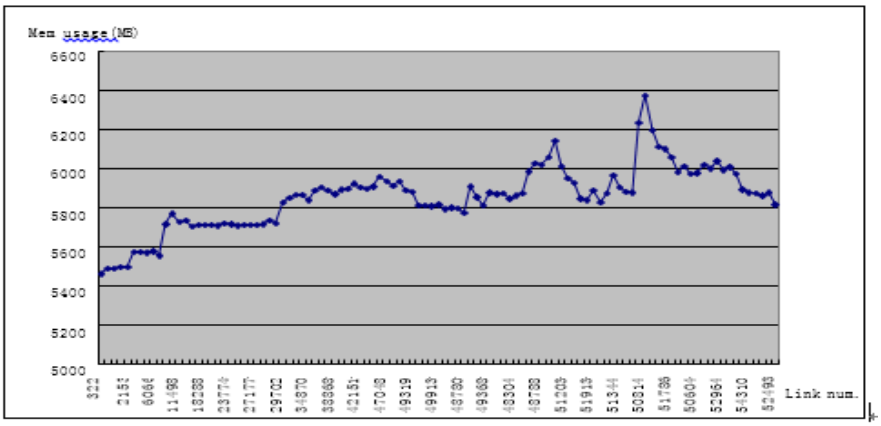

Fig. 2. Relationship between link numbers and memory utilization when accessing to common pages.

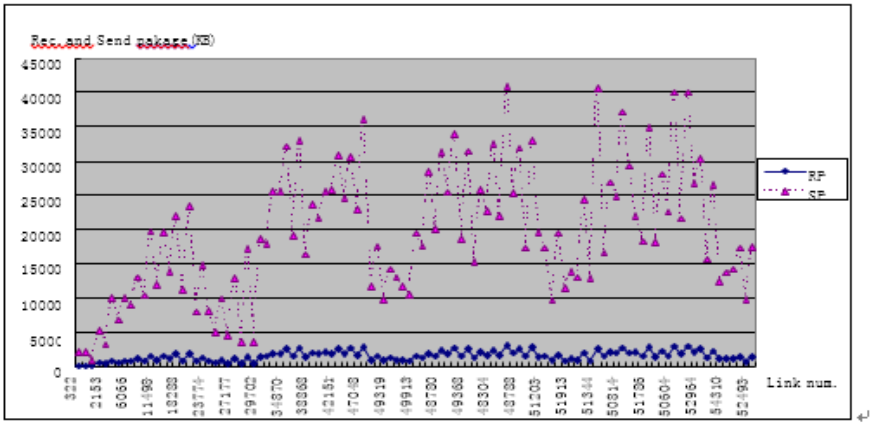

Fig. 3. Relationship between link numbers and bandwidth of receiving and sending packages when accessing to common pages

To sum up, when accessing to common webpage with small data size, busyness of system could be reflected by CPU utilization, memory utilization, link numbers and bandwidth of receiving and sending packages. It is appropriate to define and measure system loads with these indexes.

Load analysis for continuous interaction of big data size. The experimental environment for video accessing was same as above. A 500MB RMVB video was uploaded to the server and then a webpage was edited for codes of online play of this video. Similarly, information statistics were made every 5s. LoadRunner simulated that 100 users click the video page continuously. Results are shown in Fig.4, Fig.5 and Fig.6.

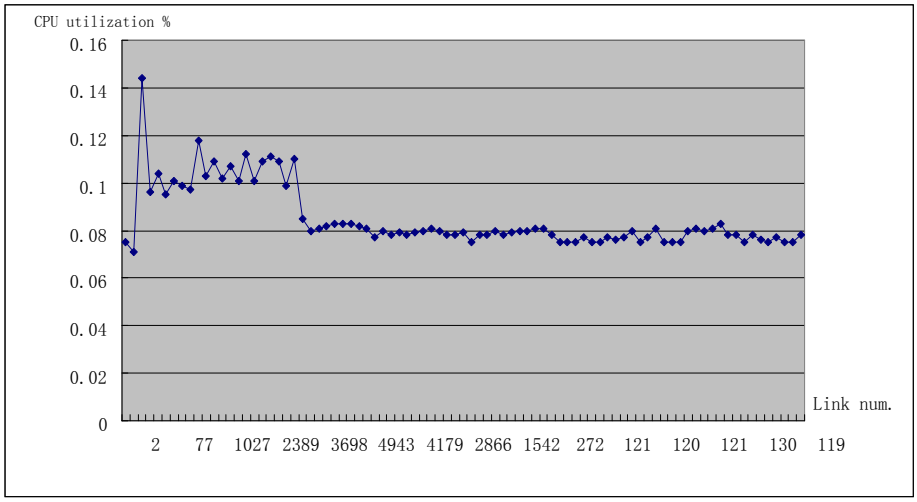

Fig. 4. Relationship between link numbers and CPU utilization when accessing to video page

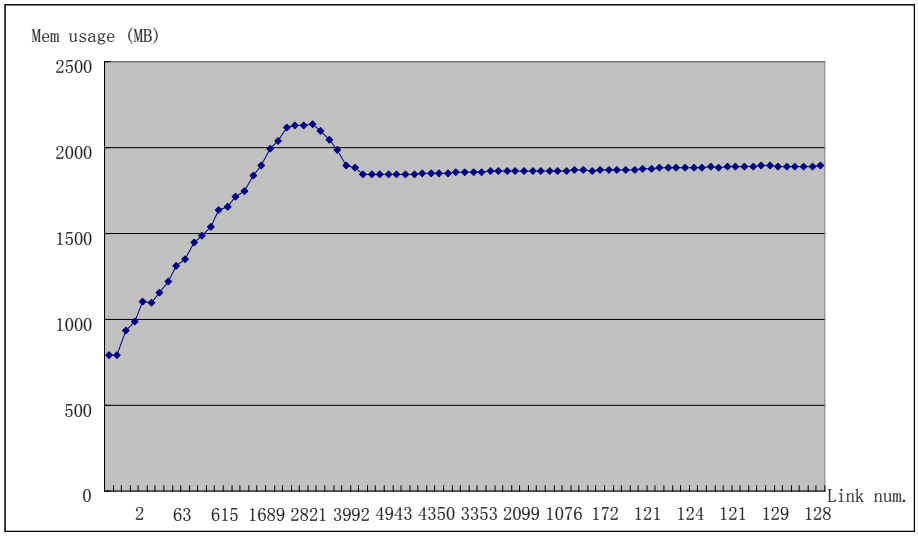

Fig. 5. Relationship between link numbers and memory utilization when accessing to video page

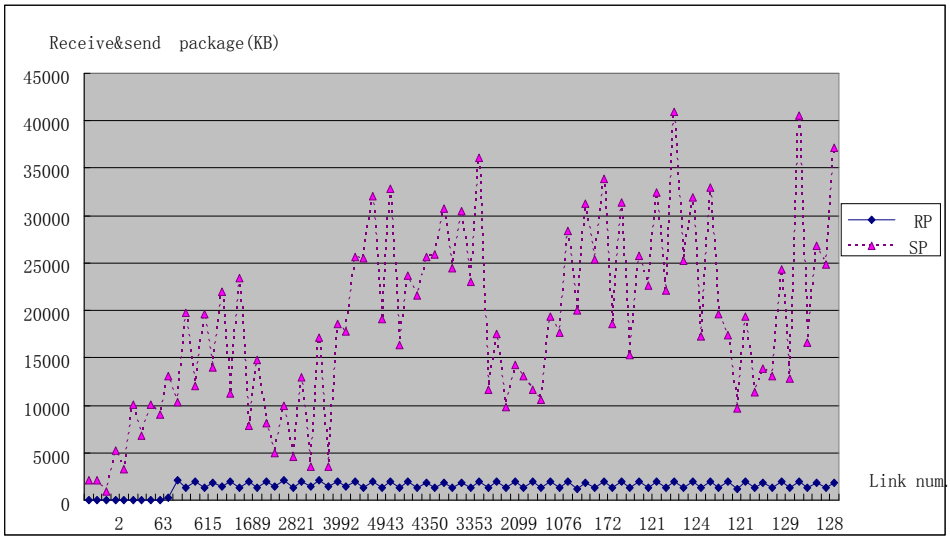

Fig. 6. Relationship between link numbers and bandwidth of receiving and sending packages when accessing to video page

In Fig.4, CPU utilization during video accessing is significantly lower than that when accessing to common pages. CPU utilization during system busy stabilizes at about 10%. It can be seen from Fig.5 that video play consumes a lot of memory. Memory utilization at accessing peak is about 1.5GB higher than that at the beginning. Fig.6 shows similar bandwidth of receiving and sending packages with that when accessing to common webpage. Additionally, the average total disk I/O during video play is 448.5, far higher than that when accessing to common webpage.

Therefore, disk I/O is an important parameter for load measurement during video play except for abovementioned CPU utilization, memory utilization, link numbers and bandwidth of receiving and sending packages.

Selection of load metrics. Load indexes are quantitative criteria of load evaluation. Different load indexes will contribute different load evaluation results at the same moment. A good load index shall have following two characteristics: (1) index data are easy to be acquired for the convenience of multiple measurements; (2) index data could reflect loading condition objectively and clearly.

Agent-based cloud mainly offers web services under the assistance of database software, cache software and web server software. According to above analysis on current internet business and two experimental results, this paper chose CPU utilization, memory utilization, bandwidth utilization of sending packages, link numbers and total disk I/O as load parameters.

In this paper, L represents server load and the five load parameters were represented by five capital letters: C for CPU utilization, M for memory utilization, P for link numbers, B for bandwidth utilization of sending packages, and D for total disk I/O.

$C, M, P, B, D \in [0,1]$. M is memory size/total physical memory size. B is bandwidth of sending packages/total bandwidth. D is the ratio between disk I/O and available I/O.

A formula used to calculate weighted sum of above parameters was proposed in Reference [23]:

$$L = \lambda_1 C + \lambda_2 M + \lambda_3 P + \lambda_4 B + \lambda_5 D \text{, and } \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 1 \quad (1)$$

In formula (1), importance of resources in the overall load is expressed by adjusting coefficients of utilization. Due to normalization of coefficients, the final calculated result is $L \in [0,1]$. The whole system will suffer bottlenecks when one resource is consumed greatly. Since the sum of coefficients in formula (1) is 1, it can only determine importance of one resource, but couldn't reflect load distribution when other resources are in severe shortage. For example, when CPU utilization is the biggest influencing factor of loading ($\lambda_1 = 0.5$), it can only show that weight coefficients of other resources are very low, but couldn't reflect the server load objectively. As a result, formula (1) neglects effect of different application on load indexes when describing server load.

Product averaging method shall be chosen when utilization of one resource exceeds the threshold and system capacity of receiving new requests declines significantly [24-25]. To describe effect of different applications on load indexes, this paper described server load by product method:

$$L = 1 - (1 - \lambda_c C) * (1 - \lambda_m M) * (1 - \lambda_p P) * (1 - \lambda_b B) * (1 - \lambda_d D) \quad (2)$$

In formula (2), $\lambda_i \geq 0$ and $0 \leq (1 - \lambda_i X_i) \leq 1$. If one application influences one or several indexes greatly, its coefficient (λ_i) can be increased accordingly. Sum of coefficients is no longer equal to 1, thus enabling to describe server load more objectively. Suppose ratios of any two load indexes are 1 at the very beginning. At one moment, network application consumes a great proportion of CPU resource and makes the server couldn't accept new tasks. Server load calculated from formula (1) isn't very high, but server load calculated from formula (2) approaches to 1. This reveals that formula (2) has stronger adaptability. It could help to increase the global load scheduling capacity, reduce flash crowd caused by normal mass access and balance loads of content access dynamically according to the processing capacity.

3.2. Variable Factor Weighted Least Connection

Weighted Least Connection considers performance difference of heterogeneous servers and servers. It takes connection load as dynamic load scheduling depression of server and assigns tasks to servers according to their weights. It gives no consideration to disk I/O and CPU utilization in real servers, thus causing serious uneven load distribution among servers under small P but high D and C.

On load balancer, P of different servers is the only one information that can be acquired in time (over frequent collection will increase loads of real servers and pressure of the central server significantly). The proposed algorithm is to adjust load distribution

strategy based on collected load information and reduce uneven loading among different servers.

In this paper, existing Weighted-Least-Connection was improved. One variable factor was added to each real server. Product of server weight and the variable factor was used as the actual weight of the server during task assignment. Specifically, after loads of real servers are collected, variable factor of the server identified with excessive loads will be reduced and the task will be reassigned using the least-connection scheduling algorithm. There are two reasons to use variable factor instead of changing server weight directly: (1) weight reflects the overall performance of one server compared to others. Positive weight represents actual service capability of the server. Therefore, direct weight changing will conflict with previous monitoring strategy and man-made strategy. (2) Weight is expressed by integral, which can only provide a poor accuracy control. On the contrary, variable factor could adjust weight flexible and contribute higher accuracy control.

Variable Factor Weighted Least Connection (VFWLC) determines that: $S=\{S_0,S_1,S_2,...,S_{n-1}\}$ is server set; m is indicator variable; i is index number of servers; $W(S_i)$ is weight of server S_i; $C(S_i)$ is current link numbers of S_i; $\alpha(S_i)$ is variable factor of S_i ($\alpha(S_i) \in [0,1]$). Servers shall be selected according to:

$$S_i = \min\left(\frac{C(S_i)}{W(S_i) * \alpha(S_i)}\right),\ i \in [0, n-1] \quad (3)$$

To accelerate operation, division was converted into multiplication. The algorithm description is shown as follows.

Algorithm 1: Variable-Factor-Weighted-Least-Connection (VFWLC)

```
Input: available server set (S), server weight set (W), variable factor set (α) and
       link numbers set of current servers (C).
Output: Chosen servers
Function VFWLC()
{
if ( W = ∅ )
   return NULL; //No server is available
m = 0;
for (i = m + 1; i< n; ++i ) {
if ( C(Sm) * W(Si) * α(Si) > C(Si) * W(Sm) * α(Sm) )
m = i;
}
C(Sm) += 1; //Plus 1 to link numbers of the server
return Sm ;
}
```

Obviously, the time complexity of the proposed VFWLC is O(n). Modification of variable factors is an important part of VFWLC. Variable factor is modified according

to server load. Load and mean load ($\overline{L}$) of servers can be calculated from formula (2) after the central server received load information of all servers at the same moment.

$$\overline{L} = \sum_{i=0}^{n-1} L_i \tag{4}$$

In this paper, server ($L_i > \overline{L}$) is recognized as overloaded. Variable factor of overloaded server will be reduced to lower its weight. If $L_i \leq \overline{L}$ and the corresponding variable factor is smaller than 1, the variable factor will be adjusted to 1 and the server will provide services according to normal weight. Other conditions won't be changes. To avoid frequent adjustment of variable factors and influence to scheduling stability, this paper set upper and lower load limits. When server load minus $\overline{L}$ exceeds φ, the server is determined as overloaded and its variable factor will be reduced. When $\overline{L}$ minus server load exceeds σ, the server is determined as under-loaded and its variable factor will be increased. The upper and lower load limits could stabilize loads of servers within a certain range.

Hence, the adjustment function of $\alpha(S_i)$ is:

$$\alpha(S_i) = \begin{cases} \overline{L} / L_i * \alpha(S_i), & L_i - \overline{L} \geq \varphi \\ 1 \quad , & \overline{L} - L_i \geq \sigma \\ \alpha(S_i) \quad , & others \end{cases} \tag{5}$$

The proposed VFWLC and adjustment function of variable factor illustrate the load balancing strategy of this paper.

4. Experimental results and analysis

This chapter made an experimental test on the proposed VFWLC and compared it with weighted Least Connection.

4.1. Experimental environment

Experimental network topologys. The experimental network topology is shown in Fig.7. Five PCs are connected directly through Gigabit switch. Client is the user computer and the IP address is 192.168.1.22. LB is load balancer which is installed with load computing module and modified load scheduling module. External virtual IP of LB is 192.168.1.100 and IP to internal server cluster is 192.168.0.100. RS1, RS2 and RS3 (Real Server 1, Real Server 2 and Real Server 3) are three cloud servers which provides real services. They are installed with load information collection module. IP of these three cloud servers are shown in Fig.7.

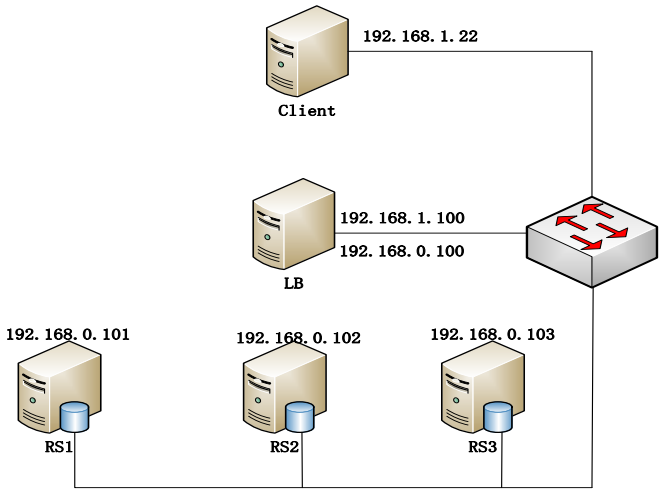

Fig. 7. Experimental network topology

Experimental environment configuration. Hardware configurations of servers are listed in Table 2. To test load balancing effect of machines with different configurations, two servers with same configuration (RS2 and RS3) and one server with poorer service capability (RS1) are used. According to their service capabilities, weight of RS1 is 7, while weights of both RS2 and RS3 are 10.

Table 2. Hardware configurations of servers

Server	Hardware configurations
Client, RS2, RS3, LB	4 AMD Opteron(tm) Processor 6136 CPU, 32G memory and 2 pieces of GB network cards
RS1	2 Intel(R) Xeon(R) E5506 CPU, 24G memory and 2 pieces of GB network cards

Software configurations of servers are presented in Table 3. To simplify CDN simulation, same web service was configured on real servers. LoadRunner was installed on the Client to simulate multiuser access.

Table 3. Software configurations of servers

Server	Software configurations
Client	Win Server2003+LoadRunner11.0
LB	CentOS 6.2
RS1, RS2 and RS3	RedHat AS5.2 +Apache+Mysql+PHP+Wordpress

In the experiment, coefficients in formula (2) were set $\{\lambda_c, \lambda_m, \lambda_p, \lambda_b, \lambda_d\} = \{0.5, 0.4, 0.4, 0.4, 0.35\}$ and two parameters in formula (5) were set $\varphi = 0.06$ and $\sigma = 0.05$.

4.2. Analysis of experimental results

Experiment design. Considering effect of different network applications on server load and scalability of the VFWLC and weighted-least-connection, this paper designed four experiments:

1) Low average load of webpage: effect of VFWLC and weighted-least-connection on server load when users visit webpage with small data size.

2) High average load of webpage: effect of VFWLC and weighted-least-connection on server load when users visit webpage with big data size.

3) Mixed webpage: effect of VFWLC and weighted-least-connection on server load when users visit webpage with small data size and webpage with big data size simultaneously.

4) Scalability test: effect of VFWLC and weighted-least-connection on server load when users visit webpage with small data size and webpage with big data size simultaneously and one additional server is involved.

LoadRunner was installed on the Client to simulate that 1,000 users visit common webpage and video webpage. LoadRunner parameters of each experiment design are shown in Table 4.

Table 4. Experimental parameters

	Number of users	Mean requests per second	Proportion of webpage with small data size	Proportion of webpage with big data size
Experiment 1	1000	1000	100%	0%
Experiment 2	1000	500	0%	100%
Experiment 3	1000	750	50%	50%
Experiment 4	1000	750	50%	50%

Result analysis. Experiment 1 simulated that 1,000 users visit common webpage continuously by using LoadRunner on the Client. Load balancing results of WLC and VFWLC are shown in Fig.8 and Fig.9, respectively. In the experiment, it took some time for request sending rate of Loadrunner reaching the determined stable value. Therefore, load of backend servers began to increase sharply at 7th information collection. It can be known from Fig.8 and Fig.9 that after 19th information collection, the request sending rate of Loadrunner stabilizes. WLC distributes load according to link numbers and weight of servers. However, different links consume different loads. The LVS system won't balance loads of servers even if it recognized uneven load distribution. Consequently, new load distribution will further intensify such uneven distribution. This could be observed in Fig.8: RS1 keeps higher load than $\overline{L}$, while RS2 and RS3 keeps

lower loads than $\overline{L}$. The proposed VFWLC will modify variable factor of servers according to their loads and redistributes loads to servers dynamically. In Fig.9, loads of three real servers are approximate to $\overline{L}$, showing slight fluctuations. VFWLC stabilizes load of servers within a certain range.

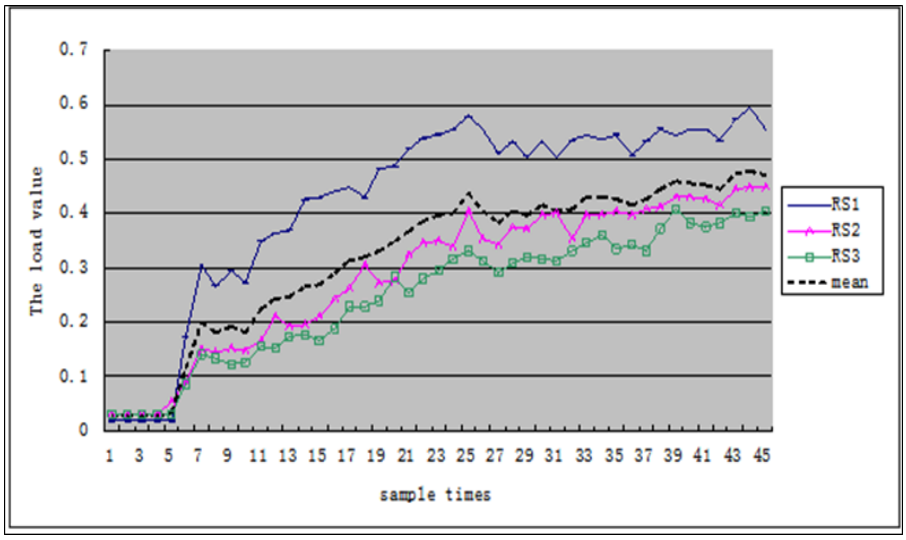

Fig.8. Load balancing result of WLC in Experiment 1

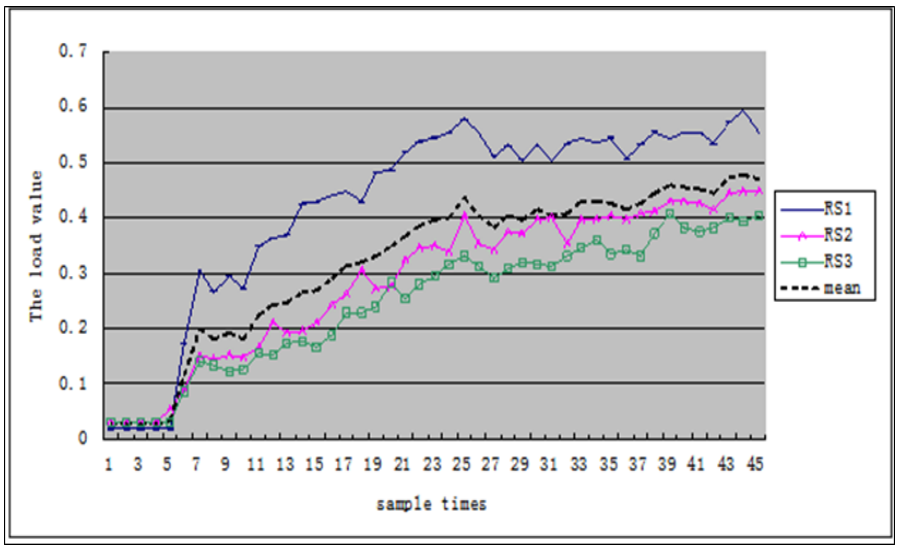

Fig.9. Load balancing result of VFWLC in Experiment 1

Experiment 2 simulated that 1,000 users visit video webpage in LVS. Load balancing results of WLC and VFWLC are shown in Fig.10 and Fig.11, respectively.

Due to the long duration of each link, link numbers could reflect loads of three real servers. Therefore, both WLC and VFWLC could balance loads of servers well. To further compare load balancing effect of WLC and VFWLC, loads calculated by WLC and VFWLC were averaged and variance sum of loads of three real servers was calculated (Fig.12). Apparently, VFWLC has significantly smaller variance sum and smaller fluctuation than WLC. This indicates that the proposed VFWLC is superior to WLC in load balancing.

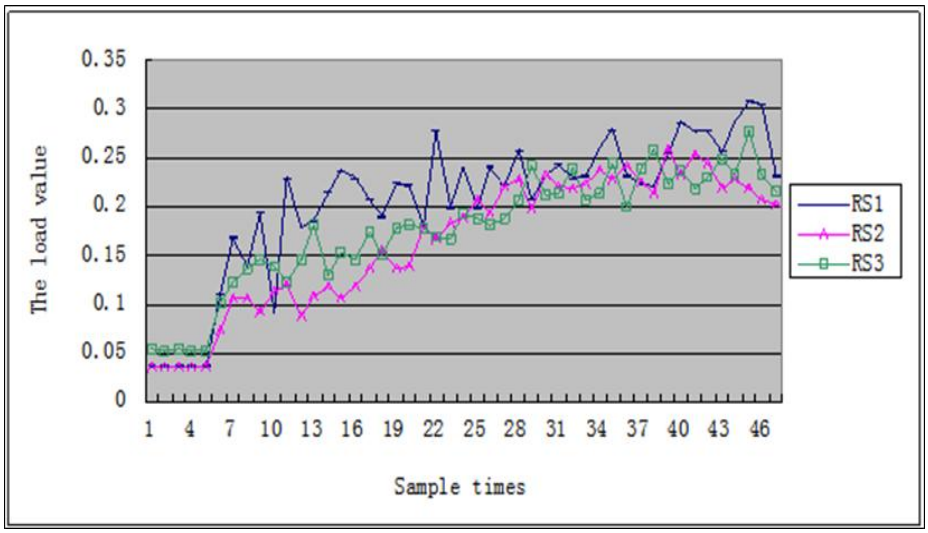

Fig. 10. Load balancing result of WLC in Experiment 2

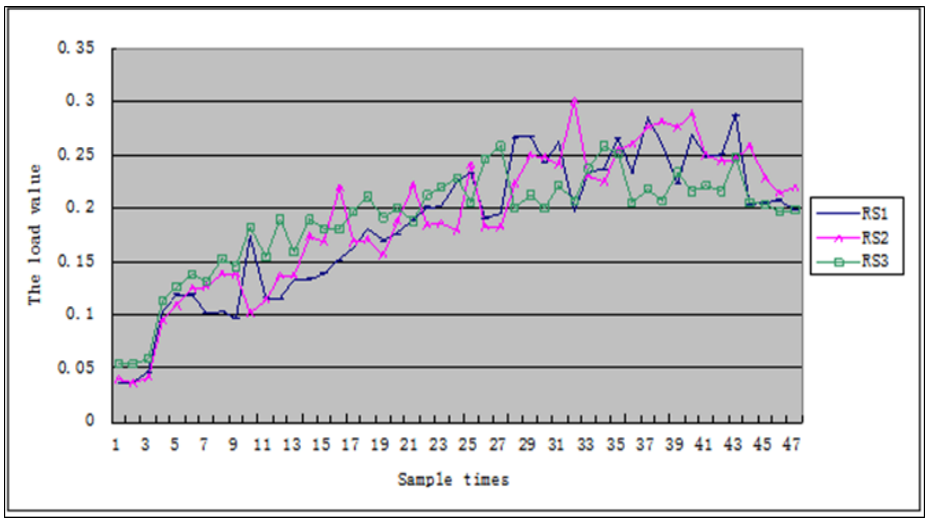

Fig. 11. Load balancing result of VFWLC in Experiment 2

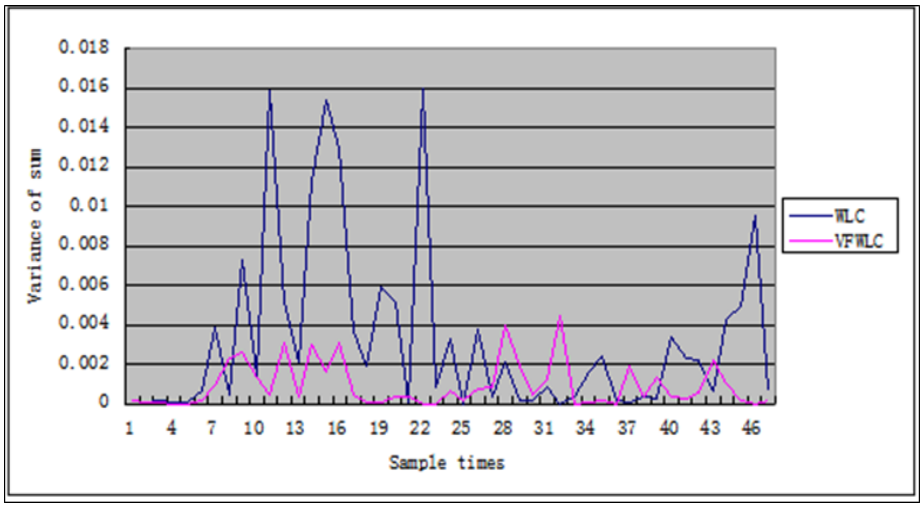

Fig. 12. Load variance sum of WLC and VFWLC in Experiment (2)

Experiment 3 simulated that 500 users visit common webpage and another 500 users visit video webpage in LVS simultaneously. Load balancing results of WLC and VFWLC are shown in Fig.13 and Fig.14, respectively. Under such mixed access, service loading of WLC fluctuates more violently than that of VFWLC. Similarly, load variance sums of WLC and VFWLC were calculated using same method mentioned above (Fig.15). According to Fig.15, VFWLC could balance loads of servers within a small fluctuation range.

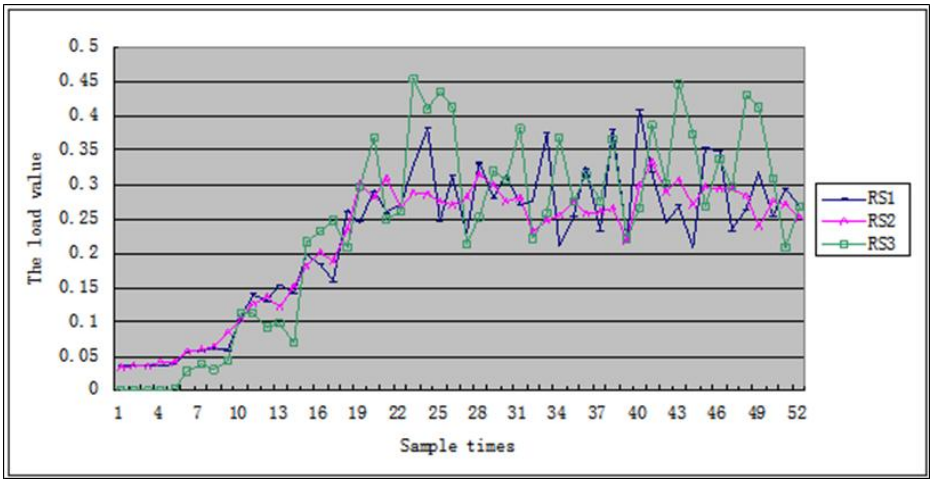

Fig. 13. Load balancing result of WLC in Experiment 3

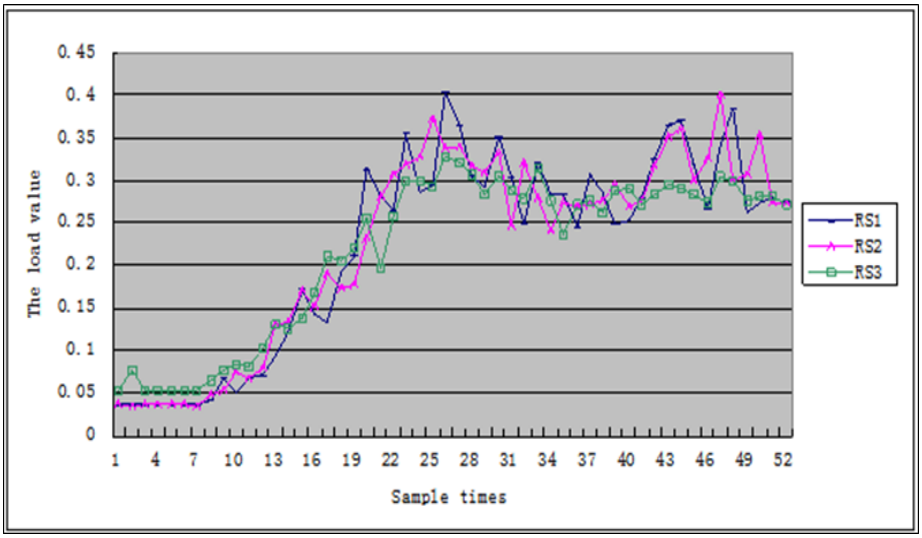

Fig. 14. Load balancing result of VFWLC in Experiment 3

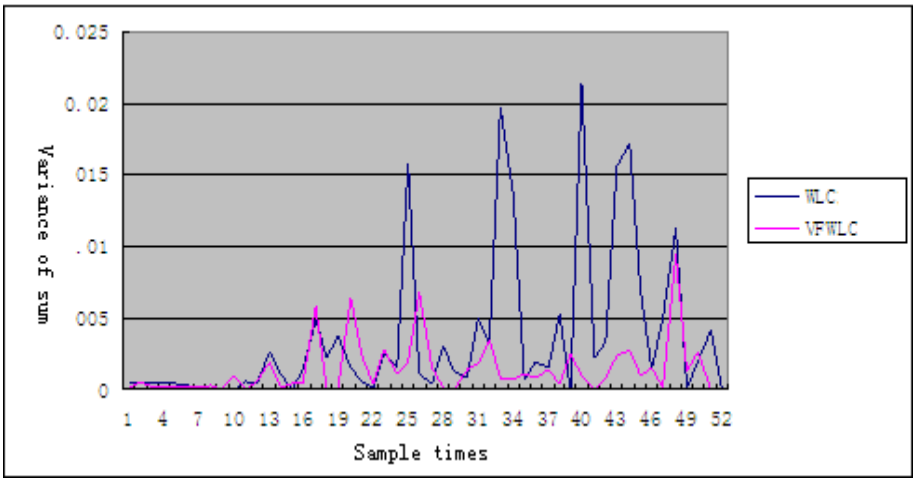

Fig. 15. Load variance sum curves of WLC and VFWLC in Experiment 3

Experiment 4 also simulated load balancing under mixed access (same with Experiment 3). It started up RS2 alone at the beginning and added RS3 in LVS when load of RS2 increases to a very high level. Load balancing results of WLC and VFWLC are shown in Fig.16 and Fig.17, respectively. Although both WLC and VFWLC could balance loads between two servers well, load balancing of WLC fluctuates greatly as time goes one, while load balancing of VFWLC fluctuates within a small range. This

implies that VFWLC could distribute load to the new added server well, contributing a stable load balancing among three servers.

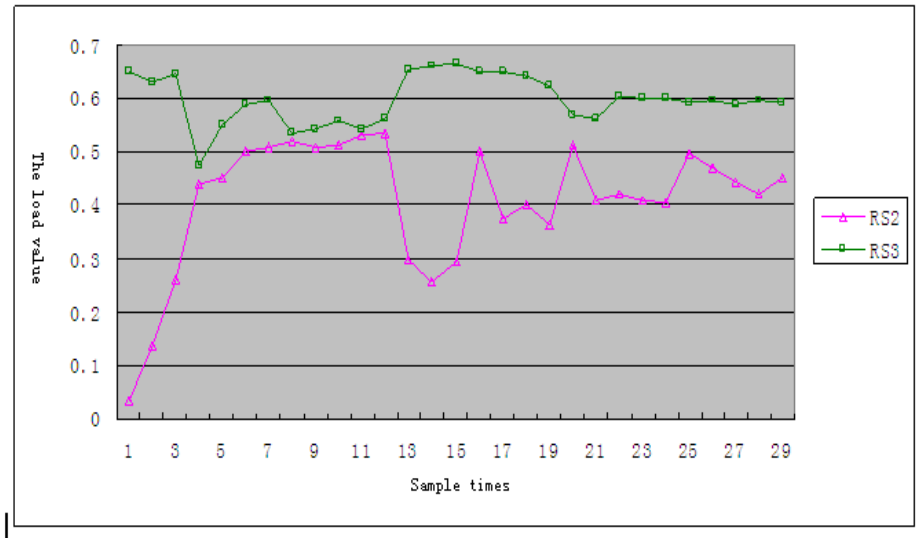

Fig. 16. Load balancing result of WLC in Experiment 4

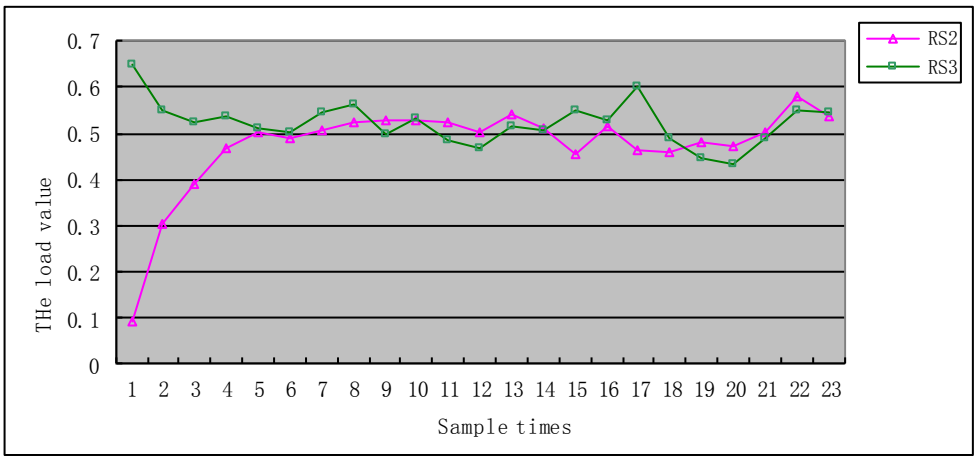

Fig. 17. Load balancing result of VFWLC in Experiment 4

To sum up, the proposed VFWLC is superior to WLC in dynamic load balancing. The proposed VFWLC could modify variable factors of servers dynamically according to load feedback of servers and make loads of all servers vary within a small range above and below the mean load.

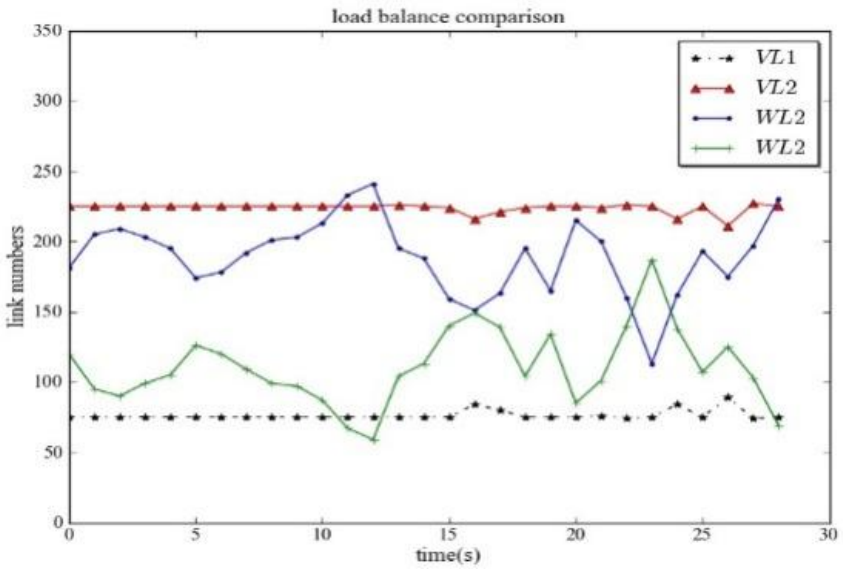

Fig. 18. Load balance comparison of servers with different service capabilities

RS2 and RS3 were further analyzed from link numbers (Fig.18). RS2 and RS3 have different performances. The proposed VFWLC (VL in Fig.18) could provide stable load balancing between them. On the contrary, load balancing of WLC fluctuates greatly. Therefore, the proposed distributed cloud brokering platform could not only provide better content services under GB network environment and high concurrent access, but also make quick responses to flash crowd.

5. Conclusions

This paper focuses on dynamic load balancing of CDN in super nodes (or clusters). Through analyzing existing load balancing system and studying scheduling strategies, a dynamic load balancing system applicable to such CDN is designed and implemented, aiming to provide high-efficiency and high-quality content services.

This paper gets some achievements:

1. Based on experiments and analysis of CDN services, this paper selects appropriate load metrics and proposes the VFWLC. VFWLC could adjust request distribution dynamically and is applicable to various network applications.

2. A CDN dynamic load balancing system based on LVS is designed and implemented on the basis of the proposed VFWLC. The proposed VFWLC is confirmed by experiments superior to classical WLC. It could balance loads among servers well and make loads of all servers fluctuate within a small range.

Future research will focus on autonomation of CDN server. Autonomic learning and machine learning shall be involved to improve adaptive ability of global load balancing strategy for cloud-oriented CDN.

Acknowledgment. This work was supported in part by the National Basic Research Program of China under Grant No. G2011CB302605. This work is partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 61173145, 61472108.

References

1. China Internet Network Information Center. Basic Research Data on China Internet Development.http://www.cnnic.net.cn/hlwfzyj/jcsc2014/201410/t20141008_49231.htm, 2014-06-30. (2014)
2. Informa Telecom & Media. Content delivery networks: Market dynamics and growth perspectives. http://www.informatandm.com/ wp-content/uploads/2012/10/CDN-whitepaper.pdf.(2012)
3. Pallis G. and Vakali A.: Insight and Perspectives for Content Delivery Networks. Communications of the ACM, Vol. 49, No.1,101-106.(2006)
4. Krishnan R., Madhyastha H., Srinivasan S., et. al.: Moving Beyond End-to-end Path Information to Optimize CDN Performance. Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. Chicago, USA, 190-201. (2009)
5. Jiang W., Rui Z. S., Rexford J. and Chiang M.: Cooperative Content Distribution and Traffic Engineering in an ISP Network. ACM SIGMETRICS Performance Evaluation Review, Vol.37,No.1, 239-250.(2009)
6. Aditya P., Zhao M., Lin Y., et. al.: Reliable Client Accounting for Hybrid Content-Distribution Networks. Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 34-45.(2012)
7. Yun B., Bo J., Jixiang Z., Qiangguo P.: An Efficient Load Balancing Technology in CDN. 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 510-514.(2009)
8. Bo J., Jixiang Z.: CDN Load Balancing Technology based on Distributed Binning Strategy. Proceeding of 5th Academic Annual Conference of China Institute of Communications, 1237-1241.(2008)
9. Yun B., Bo J.: CDN Load Balancing Technology based on Distributed Binning Strategy. Journal of University of Science and Technology of Suzhou, Vol.12, No.25, 55-59.(2008)
10. Guomin Z., Ming C., Ke D., Na W.: Distributed CDN Load Balancing Strategy based on Performance Measurement. Journal of Applied Sciences, Vol.5, No.25, 247-251.(2007)
11. Tiannan. Z.: Research on CDN Load Balancing System based on IPV6. New Technology & New Product of China, Vol.2009, No.24, 49-49.(2009)
12. Guomin Z., Jushi S., Song W., Na W.: Load Balancing Strategy for Domain-based Streaming Media CDN. New Progress of Communication Theory and Technology, 469-473.(2005)
13. Brighten G., Karthik L., Sonesh S., Richard K., Ion S.: Load Balancing in Dynamic Structured P2P Systems. IEEE INFOCOM. (2004)
14. Binjie Z., Ke X., Renjie P.: A Lookup Algorithm for P2P-CDNs. Proceedings 2010 IEEE 2nd Symposium on Web Society, 494-496. (2010)
15. Onur D.: Hydrodynamic Based Hybrid Dynamic Load Balancing, 2008 IEEE, 1-6. (2008)
16. Yan C., Xiaochun H., Taoshen L.: CDN Load Balancing Algorithm based on Air Pressure Model. Microcomputer Information, Vol.2008, No.24, 93-95. (2008)

17. Ralf D., Andreas F., Burkhard M.: Efficient schemes for nearest neighbor loadbalancing. Parallel Computing, Vol.1999, No.25:789-812.(1999)
18. Sagar D., Majeed M., Hayatorge E., Cundong Y., David A. B.: Dynamic Load Balancing in DistributedSystems in the Presence of Delays, A Regeneration-Theory Approach. IEEE Transaction on Parallel and Distributed System, Vol.4, No.18, 485-497.(2007)
19. Omp.D. P., Emerald C., Yennun H.: ONE-IP: Techniques for Hosting a Service on a Cluster of Machines. http://www.ra.ethz.ch/cdstore/www6/technical/Paper196/PAPER196.html, 1996-02-05.(1996)
20. Vittorio M., Roberto C., Andreas M., Gareth T.: Next Generation CDN servicesfor Community Networks. IEEE Coumpter Society, 89-94.(2009)
21. Wenzheng L., Qiao G., Weimin G.: Internet Load and Traffic Balancing. Journal of Shanghai University, Vol.9, No.2, 143-146.(2005)
22. Wikipedia. HP LoadRunner. http://en.wikipedia.org/wiki/LoadRunner, 2012-04-23.(2012)
23. Jingbo L., Zhila C., Anfeng L.: A Dynamic Regulation Program based on Weight of LVS Server. Computer Engineering, Vol.32, No.14, 104-106.(2006)
24. Quansheng G., Jiwu S., Xiping M., et al.: Dynamic Load Balancing Design and Implementation based on LVS system. Journal of Computer Research and Development, Vol.41, No.6, 923-929.(2004)
25. Jinpeng W., Longfa P., Jianglong L..: Dynamic Feedback Scheduling Algorithm in LVS Cluster. Computer Engineering, Vol.31, No.19,40-42.(2005)

Hui He is currently an associate professor of network security center in the Department of Computer Science, China. She received the Ph.D. in department of computer science from the Harbin Institute of Technology, China. Her research interests are mainly focused on network security, distributed computing and big data analysis. Contact her at hehui@hit.edu.cn.

Yana Feng received the M.Sc. in software engineering from the Harbin Institute of Technology, China. Her research interests are mainly focused on software engineering and software testing etc.

Zhigang Li received the M.Sc. in software engineering from the Harbin Institute of Technology, China. His main research interest is in network engineering, software security and software testing.

Zhenguang Zhu received the M.Sc. in information security from the Harbin Institute of Technology, China. His main research interest is in network security and distributed computing.

Weizhe Zhang is corresponding author of this paper, who is the professor of Harbin Institute of Technology. He has been a visiting scholar in the Department of Computer Science at University of Illinois at Urbana-Champaign and University of Houston, USA. His research interests are primarily in parallel computing, distributed computing, cloud computing. He has published more than 100 academic papers in journals, books, and conference proceedings. He is a member of the IEEE. Contact him at wzzhang@hit.edu.cn.

Albert Cheng Albert M.K. Cheng is Professor and former interim Associate Chair of the Computer Science Department at the University of Houston. He received the B.A. with Highest Honors in Comput-er Science, graduating Phi Beta Kappa, the M.S. in Computer Science with a minor in Electrical Engineering, and the Ph.D. in Computer Science, all from The University of Texas at Austin, where he held a GTE Foundation Doctoral Fellowship. A recipient of numerous awards, Prof. Cheng is the author of the popular textbook entitled Real-Time Systems: Scheduling, Analysis, and Verification (Wiley) and over 200 refereed publications on real-time, embedded, and cyber-physical systems. He is a Senior Member of the IEEE and a Fellow of the Institute of Physics.

Received: November 4, 2014; Accepted: May 19, 2015.