An Efficient Load Balancing Algorithm for Cluster System

Chunkyun Youn[1,*] and Ilyoung Chung[2]

[1] Department of Internet Software, Honam University, Kwangju, Korea
chqyoun@honam.ac.kr
[2] Department of Computer Science, Chosun University, Kwangju, Korea
iyc@mail.chosun.ac.kr

Abstract. Load balancing is one of the best efficient methods for performance improvement of cluster system. Recently, WLC algorithm is used for the load balancing of cluster system. But, the algorithm also has load imbalance between servers, because it uses inaccurate static load status of servers. In this paper, I suggest a more efficient dynamic load balancing algorithm base on various load status information of servers by real time. It shows that load imbalance phenomenon is improved greatly and response time is also improved compare with WLC algorithm.

1 Introductions

Fast growing Internet user and huge amount of multimedia data are rapidly increasing network traffic. Servers and network are bottle-neck in this situation. Now a days, performance elevation and high availability of server are important to solve the problem [1]. Various cluster systems are used as suitable solution of it [2, 3]. Among them, load sharing cluster system consists of several low-cost servers which are connected to high speed network, and applies load balancing technique between servers. It offers high computing power and high availability.

The load balancing algorithm is core function of the cluster system. Many techniques were studied. Well known algorithms are round-robin (RR) scheduling [4], weighted round-robin (WRR) scheduling [5], least-connection (LC) scheduling [6] and WLC (Weighted Least Connection) scheduling [7]. The WLC is widely used now among them.

Above load balancing algorithms select a server according to fixed weights which are calculated by server's physical processing capacity and the number of established connections mainly. Such methods can't know server's load state exactly, because those are not considered various load elements of real servers. And measuring time is not suitable, because Director gets the connection number of real servers periodically. So, it is not correct load of real servers. That is, inaccurate load status and unsuitable measuring time are the cause of load imbalance.

* Corresponding author.

H. Jin, D. Reed, and W. Jiang (Eds.): NPC 2005, LNCS 3779, pp. 176 – 179, 2005.

2 Proposal of an Efficient Load Balancing Algorithm

2.1 Various Load Elements Investigation and Application Plan

In this paper, various load elements of UNIX web server are considered to measure exact load situation. CPU, memory and network are selected as influential suitable elements among them. The detail statuses of main load elements are followings;

CPU load. Usually, we have to collect whole CPU usage, average CPU load and CPU usage of each process etc to measure CPU load. When a client requests connection, correct present CPU load of real servers is very important to decide which server will handle the request. Numbers of waiting process is suitable for that purpose. It can be different according to cluster system configuration, number of users and concurrent connection ratio etc. Usually, connection requests are processed without waiting because servers are very powerful. Therefore, if there is waiting processes that mean the CPU is busy. So, we can select which server has lower load [8, 9].

Load of memory. We can use virtual memory amount of processes, free memory amount and paging activity that are performed in the latest 20 seconds from memory. We can confirm relatively exact present memory load by the free memory amount among them [8, 9].

Load of network. Packet I/O amount of each network interface, packet error rate and collision rate are available for load status of network. We can estimate that a network interface is over load if collision rate approaches to 5 ~ 10%, and use packet I/O amount if necessary [8, 9].

2.2 Dynamic Load Measuring and Balancing Algorithm

I propose a dynamic load measuring algorithm (Fig. 1) that can collect load status of server base on the selected elements by real time. It will be loaded on each real server and called using broadcasting RPC by Director. A called real server collects own load status according to Fig. 1 algorithm and transmits it to the director. The value "Y" and "Init_Average" should be adjusted properly according to configuration of cluster system and users' environment after system configuration.

Fig. 3 shows the proposed load balancing algorithm that handles user's request with real time load status of servers.

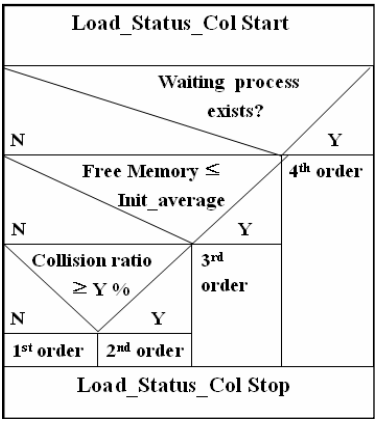

Fig. 1. Load measuring algorithm

Fig. 2 and 3 show the proposed prototype modules configuration and load balancing algorithm.

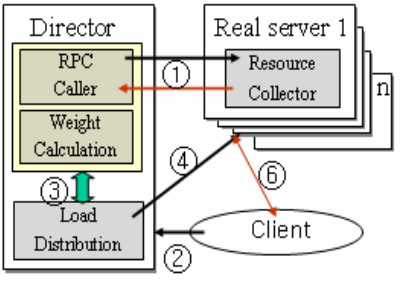

Fig. 2. Prototype module configuration

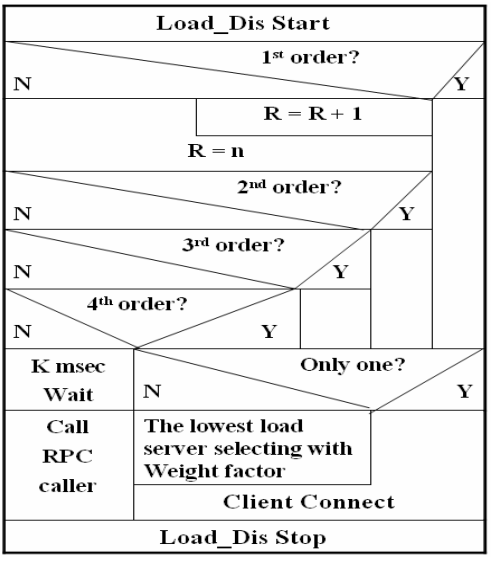

Fig. 3. Load balancing algorithm

3 Test and Results Analysis

I use the WLC which is the most efficient among existing algorithms for performance comparative test of the proposed algorithm. Comparison items are free memory change of each real server and response time of cluster system for the two algorithms.

3.1 Test Result Analysis for Free Memory

When number of concurrent connecters is below 200, free memory difference of each server is not so big in the WLC and the proposed algorithm. But, when the number is

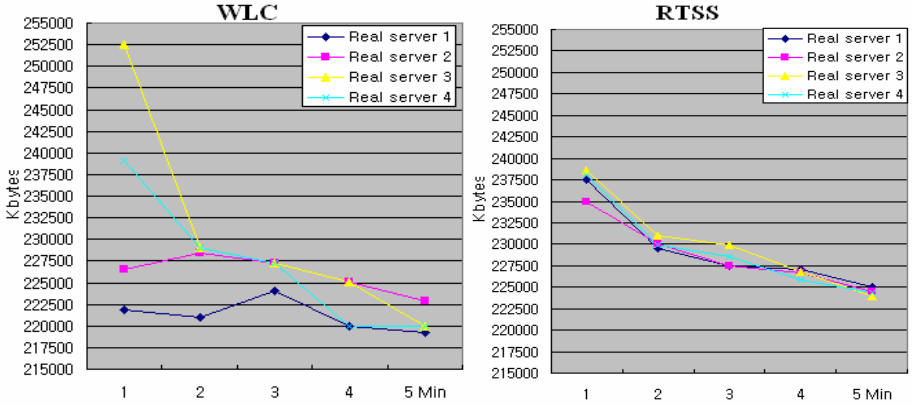

Fig. 4. Free memory changes of WLC and the proposed algorithm (at 400 numbers)

400, server's free memory of the WLC is not even, while it is similar in the proposed algorithm (RTSS) as shown Fig. 4. This means that more efficient load balancing was done by the proposed algorithm.

3.2 Test Result Analysis for Response Time

Fig. 5 shows the test result for average response time of two algorithms by the number of concurrent connecters. Response time of the proposed algorithm (RTSS) is improved 9.3msec than existing algorithm (WLC) in case of 100, while it is improved 203msec in case of 400.

When the number of concurrent connecter is few, the response time is not so big different. But, when it is increased, the difference is big. This means performance of cluster system is optimized well in the proposed algorithm.

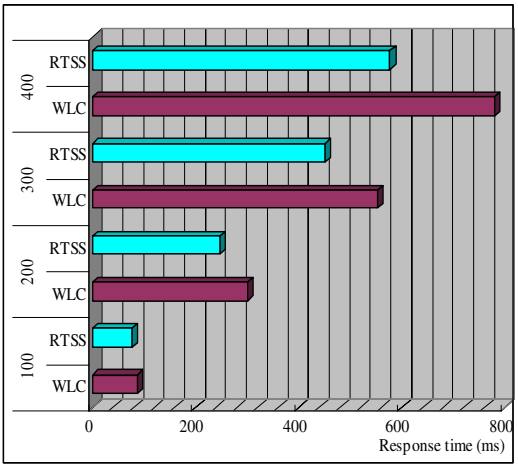

Fig. 5. Results of response time comparison

4 Conclusions

I proposed an efficient load balancing algorithm to improve the performance of cluster system. The WLC algorithm tries to balance load according to the fixed physical resources of real servers' and connection numbers. On the other hand, the proposed algorithm measures waiting process, free memory and collision rate by real time to get more accurate load state of real servers, and used them to balance load efficiently.

References

1. Delivering High Availability Solutions with Red Hat Enterprise Linux AS 2.1, RedHat (2003)
2. Jian liu, Lorghu Xu, Baogen Gu, Jing Zhang, A scalable, high performance Internet cluster server, High performance computing in the Asia-Pacific region, 2000 Proceedings. The firth International Conference/ Exhibition, Vol.2, (2000) 941-944
3. OYoung Kwon, Cluster system introduction Korea institute of science and technology information news letter (2000)