Dynamic load balancing scheme on massive file transfer system

Hou Weiguang[1,a], He Gang[2] and Liu Xinwen[3]

[1]Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Beijing University of Posts and Telecommunications, Beijing 100876, China
[3]Wisetone Technologies, Beijing 100876, China

Abstract. In this paper, a dynamic load balancing scheme applied to massive file transfer system is proposed. The scheme is designed to load balance FTP server cluster. Instead of recording connection number, runtime load information of each server is periodically collected and used in combination of static performance parameters collected on server startup to calculate the weight of servers. Improved Weighted Round-Robin algorithm is adopted in this scheme. Importantly, the weight of each server is initialized with static performance parameters and dynamically modified according to the runtime load. Apache Zookeeper cluster is used to receive all information and it will inform director of the runtime load variation and offline behavior of any server. In order to evaluate the effect of this scheme, a contrast experiment with LVS is also conducted.

1 Introduction

The rapid growth of Internet makes the number of access to multimedia network servers increase rapidly. The server needs to provide service to a large number of concurrent accesses. The ability of processing and I/O has become the bottleneck of providing services. To solve this problem, one solution is expensive high-performance server or SMP, the other is connecting multiple servers to form a cluster so that performance could be improved through parallel processing and high speed information exchange between each other. The latter solution, with high overall performance (such as response time, throughput), high scalability, high availability, and higher performance/price ratio, has become the principal method to build high performance information servers.

Load balancing is the core part of the normal work in cluster system. Its main purpose is to distribute tasks reasonably to all of the nodes in cluster, achieve the balanced state of the whole system and ensure the processing capacity and quality of service of the system. Load balancing can be achieved directly based on hardware products or software[1]. The load balancer is installed between the server and the external network. The hardware implementation of the load controller is expensive and not flexible, and it couldn't support more improved load balancing strategy and more complex application protocol.

In software scheme, the load balancing algorithms could be classified into static and dynamic algorithm. Studies have shown that both the static load balancing scheduling and the dynamic load balancing algorithms could improve the performance of the cluster system. But in practice both of the scheduling methods still have some limitations.

The static scheduling is poor in adaptability. It never takes the cluster's runtime state into consideration for the directing scheme is fixed. While the parameters traditional dynamic scheduling utilized like connections number couldn't precisely reflect the load condition[2].

The dynamic load balancing scheme proposed in this paper is capable of balancing nodes in cluster on the basis of runtime load information and it also takes the hardware static performance into consideration. It has great adaptability, working well in heterogeneous cluster and makes self-adjustment in time when load condition of cluster changes.

2 System model

The cluster system is consisted of a director and several file servers that connected by a high-speed network.

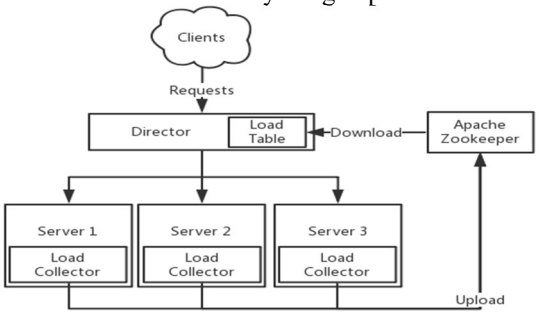

Figure 1. System Model

More details of the system model are not described in this figure. Important features left are as follows:

1) There is a load collector on every file server, collect static performance parameters on server startup and periodically collecting runtime load information.

2) An Apache Zookeeper cluster works as the manager of load information. All load collectors on server

* Corresponding author: [a] hwgonly1@bupt.edu.cn

have registered a corresponding znode in zookeeper and upload load information to it.

3) Director has registered child-watcher on servers' root znode and data-watcher every server's znode that stores load data to monitor all servers' state.

At time to refresh the load, load collector collects load information and update the corresponding zookeeper node. Then the director is informed and refresh the load table. Weights of servers are recalculated on basis of the load table.

3 Znode structure

As a commonly used distributed coordination service provider, Apache Zookeeper could work as a data manager and it guarantees data consistency[3]. File system in Zookeeper is organized as hierarchy tree structure and every node in it is called znode.

There are two types of znode that a client can create[4]. Persistent—clients manipulate create and deleting them explicitly. Ephemeral—client create such znode and they either delete them explicitly or let the system remove them automatically when the session that creates them terminates.

Based on znode's feature, static performance parameters and runtime load information are stored at different znodes as figure 2 shows.

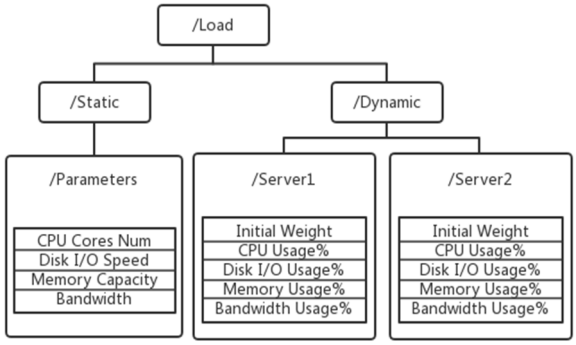

Figure 2. Znode Structure

3.1.Static performance parameters

Static parameters are used to measure the carrying capability of server and initialize the weight of server. It includes number of CPU cores, disk I/O speed, memory capacity and bandwidth. At server startup, load collector collects all these parameters, store them into serialized data structure. After data collection, load collector will communicate with zookeeper cluster to examine whether "Parameters" znode exists under "Static" znode. If it doesn't exist, then the Zookeeper client on this server will create it with the static performance parameters and record the current server's weight as 1. Otherwise, the Zookeeper client will get the data "Parameters" znode records and calculate current server's weight and record the weight locally.

3.2 Runtime Load Information

Znodes stored with runtime load information are ephemeral znodes. Servers' online/offline behavior will lead to the creation and elimination of corresponding child znode under "Dynamic" znode. Therefore, director has registered child listener to "Dynamic" znode to be informed of online/offline and data listener to every server's corresponding znode to be informed of the load change[5].

Runtime load information is used to dynamically modify the weight of server. It includes initial weight, usage percentage of CPU, disk I/O, memory and bandwidth. These information is periodically collected and upload to the corresponding znode that attached to "Dynamic" znode. Every time runtime load information updated the director will be informed.

4 Load balancing algorithm

4.1.Basic algorithm

Servers' runtime usage percentage of hardware resources are utilized to measure load condition rather than recording connection number in this scheme. Therefore, the commonly used algorithm Improved Weighted Round-Robin is the most suitable one to be applied in this scheme.

Assume there are three file servers in cluster and corresponding weights are W_1, W_2 and W_3. The implementation of Improved WRR needs a new data structure "Node" consists of id, weight and current weight of server. The calculation of the server list to be allocated is described in pseudocode as follows.

Table 1. Improved WRR Pseudocode

```
Input:    list of Nodes
Output:   list of servers to be allocated
length=W1+W2+W3
for i in 1 to length
    maxWeight=0
    loc=Input[0]
    for node in Input
        if maxWeight <node.currentWeight
            then loc=Node
        Node.currentWeight+=Node.weight
    end
    loc.currentWeight-=length
    Output[i]=loc
end
```

Compared with traditional WRR, Improved WRR is more suitable to work as the basis of dynamic load balancing scheme. Traditional WRR does not scatter elements of high weight in server list when selecting element. Therefore, it will give great pressure to servers with higher weight. On the contrary, Improved WRR scatters elements in server list so that servers will be alternately selected and the weight will be considered at the same time[6]. This mechanism will lead to better balance effect.

4.2.Weight initialization

At server start-up, every server collects its own static performance parameters and download the standard parameters stored in "/Load/Static/Parameters" znode if the node exists. Then weighted summation is used to calculate the initial weight. All parameters' definition concerning initial weight calculation are presented in table 2.

Table 2. Parameters' Definition

Parameters	Standard	Server	Weight
CPU Cores Number	C_0	C	W_c
Disk I/O speed	D_0	D	W_d
Memory Capacity	M_0	M	W_m
Bandwidth	B_0	B	W_b

Define the initial weight as W_i, weighted summation is used to calculate.

$$W_i = W_c * \left(\frac{C}{C_0}\right) + W_d * \left(\frac{D}{D_0}\right) + W_m * \left(\frac{M}{M_0}\right) + W_b * \left(\frac{B}{B_0}\right) \quad (1)$$

In FTP cluster, the usage of bandwidth is beyond the others. After adjusted for many times, weights in this equation [W_c, W_d,W_m,W_b] is set to be [0.2,0.1,0.1,0.6].

4.3.Weight dynamic modification

4.3.1. Calculation of load factor

The load collector running on server periodically collects runtime load information. This information will be updated to corresponding znode and the director will use it to calculate load factor F. The load factor is also calculated through a weighted calculation formula. All parameters related to dynamic load factor are defined in table 3.

Table 3. Parameters' Definition

Parameters	Server	Weight
CPU Usage	C	W_c
Disk I/O Occupancy	D	W_d
Memory Allocation	M	W_m
Bandwidth Occupancy	B	W_b

The calculation of dynamic load factor is stated as follows.

$$F = 1 - (1 - W_c * C) * (1 - W_d * D) * (1 - W_m * M) * (1 - W_b * B) \quad (2)$$

This formula is chosen rather than weighted summation for it is more efficient in detecting heavy load on a particular hardware. If usage of single hardware resource is much higher than others, then the load factor should be nearly 1 to warn that the server is very busy but weighted summation couldn't work out. This formula suits the requirement better than weighted summation.

All these weights to calculate the factor can be modified to satisfy special requirements such as setting maximum usage of a particular hardware to make sure it's always able to deal with another task. In experimental environment there is no special requirement so [W_c, W_d,W_m,W_b] is set to be [1,1,1,1].

4.3.2. Weight modification

Every time load factor is updated, the weight server is also calculated. Define the modified weight as W_m then the modification of weight is stated as follows:

$$W_m = \text{Math.around}[(1 - F) * W_i * E] \quad (3)$$

E in this equation is used to amplify the weight. Both (1-F) and W_i are smaller float than 1 but the weight applied into Improved WRR needs to be integer so it's necessary to set an amplify factor. In experiment it's set E=10. And the final result will be converted to integer.

In case all servers in cluster are busy and W_m is below or equal to 0, W_m will be set to 1 if the final result is 0. This strategy guarantees that there will always be available server to be allocated.

4.4.Re-allocate interval

Ftp cluster provides service to a large number of ftp clients. If client requests for server allocation every time establishing connection, the pressure on director will be increasing rapidly. In this load balancing scheme ftp client requests for re-allocation at set intervals rather than every time. The time interval is returned by director along with the server address allocated. It's also modified according to the load balancing condition. Assume the load factor of servers in cluster is [F_1,F_2,F_3], the ceiling interval, floor interval and initial interval are [I_c,I_f,I_i].The calculation logic of the time interval in pseudocode is stated as follows.

Table 4. Time Interval Calculation

```
Interval=Ii
max=0, min=1
for f in [F1,F2,F3]
    if f<min
        min=f
    elsif f>max
        max=f
end
if max-min>threshold
    Interval=max(interval/2,If)
else
    Interval=min(interval*2,Ic)
```

The overall logic of time interval modification is to increase if the balance condition is good and decrease on the contrary. It helps to balance the load in time and reduce the pressure on director.

5 Contrast experiment

To test the performance of dynamic load balancing scheme, a contrast experiment with Linux Virtual Server is conducted.

5.1.Experimental environment

5.1.1. Dynamic load balancing

The experiment is conducted in a cluster with 7 servers in it. One server works as director, three servers work as ftp server and the rest work as ftp clients. All these servers' hardware configuration is the same.

Table 5. Servers' Task

Server Role	Task
Director	Director Program Load Information Recorder Apache Zookeeper in standalone mode
FTP Servers	Server Program VSFTP
FTP Clients	Concurrent FTP Client

In experiment, load information that collector uploaded is also recorded into files. Compared to connection number and response delay, load information is better to be used to evaluate the balance effect. Gap between different servers' usage percent directly represents balance effect.

Server works as ftp client will launch multiple threads and start an ftp client in every thread to simulate a large number of clients.

5.1.2. LVS

As comparison, the experimental environment is set similar to the environment mentioned above.

Table 6. Servers' Task

Server Role	Task
Director	Ipvsadm Load Information Recorder
FTP Servers	VSFTP
FTP Clients	Concurrent FTP Client

LVS has already been integrated into Linux kernel so only the administration program—Ipvsadm has to be launched on director[7]. Commonly used algorithm—SED(Shortest Expected Delay Scheduling) is chosen in experiment.

File transition must be finished in FTP PASV mode. Only in this mode, director doesn't need to forward all the packets so that its performance won't be the bottleneck of cluster. Iptables' rules are also added on every file server to solve the address conflict in PASV mode[8].

5.2.Experimental scheme

The main experimental scheme is to test balance effect under different level of load pressure. As max upload speed of single client is configured to fixed value, the number of clients is proportional to the load pressure. The purpose of the experiment can be achieved by adjusting the number of concurrent clients.

Extra load pressure is also added to servers by executing additional tasks on these servers. Through experimental observation, it's known that bandwidth usage percent is much higher than the others. Taken bandwidth usage as the main observation object, additional file transmission of other protocol is the best way could add extra load to server. In experiment multi-thread SCP clients are launched[9] on one of the servers running FTP clients and transfer files to one particular FTP server to supply extra load.

5.3.Experiment result analyzation

The line chart of bandwidth usage is chosen to be analyzed because the usage of bandwidth is much higher than other hardware.

In contrast experiment, the duration of each experiment is about 10 minutes, and the first 3-4 minutes for the FTP balance effect test. After that, additional load is added and the load changes are observed. After optimizing the weighting coefficients by several experiments, the test results show that the trend of bandwidth occupancy rate of FTP servers in each platform is the same under the different number of FTP clients. The test results of 90 concurrent clients transmitting files are analyzed.

Figure 3. Bandwidth Usage of LVS SED

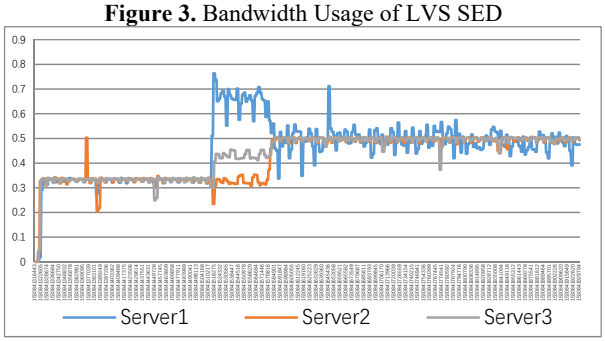

Figure 4. Bandwidth Usage of Dynamic LB Scheme

As can be seen in line charts, in first 3-4 minutes the bandwidth usage curves of different servers under dynamic LB scheme almost coincide with each other. This trend is identical with LVS in SED mode, proving the dynamic scheme could achieve similar balance effect to traditional balance scheme.

After extra load is added, the bandwidth usage trend is different. Under LVS, number of connections is the only indicator to balance the cluster so the server with additional load will still be allocated approximately the number of FTP requests similar to other servers[10]. The consequence is that the server bears additional load will be much busier than the others.

Under dynamic scheme proposed in this paper, however, the cluster is able to self-adjust to achieve a balanced state. Figure 4 shows that about 1 minute later after the extra load is added, the other two servers have

already been allocated more requests and the cluster has achieved a new balanced state.

6 Conclusion

In practice, clusters are not only responsible for one task. Sometimes file servers are responsible for some task such as file integration or file pre-processing. Traditional load balancing schemes, such as LVS, tend to be more balanced in a particular business aspect, and cannot be adjusted according to the actual situation when server's other business loads are heavy.

The dynamic load balancing scheme proposed in this paper is proved to be more aware of cluster's actual load condition and is effective in self-adjustment. It could achieve better balance effect than LVS in practice. And because the load information collecting and weight calculation is lightweight task, the additional load it brings is negligible. Therefore, this scheme is ideal for massive file transfer system.

References

1. M. Katyal, A. Mishra. *A comparative study of load balancing algorithms in cloud computing environment.* arXiv preprint arXiv:1403.6918 (2014).
2. Yichuan Jiang. *A survey of task allocation and load balancing in distributed systems.* IEEE Transactions on Parallel and Distributed Systems 27.2:285-299.(2016).
3. Saurav Haloi. *Apache ZooKeeper Essentials*. Packt Publishing Ltd, 2015.
4. Hunt Patrick, Mahadev Konar, Flavio Paiva Junqueira, Benjamin Reed. *ZooKeeper: Wait-free Coordination for Internet-scale Systems.* USENIX annual technical conference, vol. 8, no. 9. 2010.
5. Artho, Cyrille, et al. *Model-based API testing of Apache ZooKeeper. Software Testing, Verification and Validation (ICST), 2017 IEEE International Conference on*. IEEE, 2017.
6. *Upstream: smooth weighted round-robin balancing.* https://github.com/phusion/nginx/commit/27e94984486058d73157038f7950a0a36ecc6e35
7. Wensong Zhang. *Linux Virtual Server for Scalable Network Services.* Ottawa Linux Symposium. Vol. 2000(2000).
8. Chen, Yan-sheng, et al. *Design and implementation of FTP service resource sharing platform based on LVS load balancing technology.* CSIA 2014, Bangkok, Thailand, November 17-18, 2014. CRC Press, 2015.
9. Andrew van Rooyen. *Determining the optimal protocols for transfer of large data files.* (2015).
10. Choi DongJun, Kwang Sik Chung, JinGon Shon. *An improvement on the weighted lest-coonection scheduling algorithm for load balancing in web cluster systems.* Grid and Distributed Computing, Control and Automation. Springer, Berlin, Heidelberg, 2010. 127-134.