An Improved Multimedia Conference System with Load Balance

Zhang Yundu, Shang Yanlei
The State Key Lab of Networking and Switching
Beijing University of Posts and Telecommunications
Beijing, China
yundu0819@gmail.com, shangyl@bupt.edu.cn

***Abstract*—With the development of economics and the globalization of trade, the Multimedia Conference System is now widely used and the number of its users increases exponentially. But a single server can't handle the high concurrent user requests effectively. In this paper, we improved the Multimedia Conference System with load balance technology. With the load balance technology, the robustness and availability of the system can be enhanced significantly.**

Keywords-load balance, cluster, Nginx, Multimedia Conference System

I. INTRODUCTION

With the explosive popularity of the Internet and the World Wide Web, most popular web sites confronted with a problem that the server overloaded caused by the increasing number of concurrent accesses. In order to process the user requests timely, increase the network throughput and improve the quality of service, it's necessary to upgrade the hardware and software of the sever. But the the hardware is expensive and non-scalability. In this case, server cluster system appears. The sever cluster system refers to a server group which is composed of more than one homogeneous or heterogeneous severs and can provides services that are transparent for the external users. And it becomes a key point that how to achieve a reasonable distribution of load between multiple servers and avoid appearing one server with full load while other servers with little load. In this situation, the load balance technology was born. In recent years, cluster technology and load balance technology got fully developed and they have significant effect on solving the problem of overloading.

As a web service, the Multimedia Conference System also has to face the problem that the overload of the server and uneven distribution of resources when there are too many users access the system. Using cluster and load balance technology, the requests the system received can be evenly distributed on each server, thus we can reduce the response time of the requests and improve the resource utilization.

The Multimedia Conference System proposed in this paper contains Client Layer, Load Balancer Layer, Server Cluster Layer and Media Server Layer. The Client Layer receives the requests of the user including creating a conference and joining an existing conference. The Load Balancer Layer redirect the user's request to one of the application servers in the cluster according to the load balance algorithm and the different parameters from the Client Layer. The application servers in Server Cluster Layer call the Media Server resources and return them to the Client Layer. This implements a basic process of the system. The Load Balancer Layer contains four modules including User Authentication, Decision-Making, Redirecting and Log Parsing. With the Load Balancer Layer and its load balance algorithm, the user's request can be distributed on different servers and those who want to join the same conference can be allocated to the same application server. Thus we can minimize the response time of the request, improve the resource utilization and also ensure the conference can be held correctly.

The rest of this paper is organized as follows: we describe related work in section II. Section III presents the architecture of the Multimedia Conference System. In section IV, we display the implementation of the Load Balancer Layer in detail. Some experiments of key features of load balancer in section V. We conclude in section VI.

II. RELATED WORK

Load balancing has been a hot research topic since the 1980s. After in-depth study by a large number of researchers, load balance technology got rapid developed.

Many domestic and foreign manufacturers have launched a dedicated load balancer for cluster systems. Currently mature cluster systems provided effective load balancing schemes, such as Microsoft cluster, WebSphere and so on.

In China, there are several universities studying the load balance cluster system, such as Tsinghua University, Zhejiang University and the National University of Defence Technology. And Tsinghua University implements the scalable parallel Web server cluster system – TH-Web Cluster.

May 1998, Dr. Zhang Wensong presided over the development of the Linux Virtual Server (LVS) [1] Project. The project proposed a load balance scheduling solution based on IP layer and based on the content request distribution. And now the LVS has been implemented in the Linux kernel. The project can make a group of servers constitute a virtual server which is scalable and can provide high available network services. At present, the LVS technology has been widely supported. Many well-known international large companies, such as Red Hat, Turbo Linux, Ericsson and Red Flag Software have launched some load balance software based on LVS. With the maturity of the Linux operating system, LVS's performance becomes more stable and its application scope will be more widely.

Besides the LVS which is open source, there are still two widely used open source software of load balance. One is

Tomcat-Connector [2] which is developed by the Tomcat Project team and the other is Nginx [3] which is developed by Russian researchers. They both encourage enthusiasts redevelop and optimize their basic functions.

Nginx can achieve load balance function and still can be used as a lightweight HTTP server [4]. Therefore, our system adopts Nginx to implement the load balance function. We make a brief introduction of Nginx, and the architecture of Nginx is shown in Fig. 1.

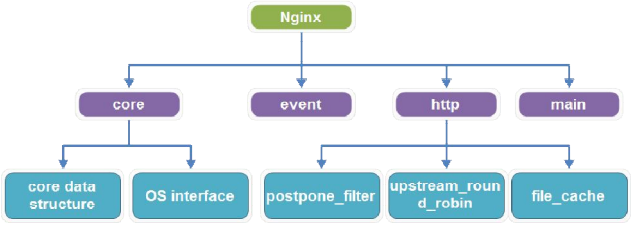

Figure 1. The Nginx Architecture Diagram

Nginx consists of four core modules: core, event, http and main. Each module also contains its expansion modules. Among them, the core module definition in the Nginx important data structure; the HTTP module defines the HTTP request and response processing API; files under the src/http/modules are http expansion module. Nginx receives the user's HTTP request and distribute them to the handler or the load balancer on the basis of the configuration file. When the request is assigned to the load balancer, it will forward the request as HTTP1.0 agreement to the backend server. After responding from the server, the load balancer will forward the response, and through several filter processing, the response will eventually returned to the user as HTTP response.

III. ARCHITECTURE OF MULTIMEDIA CONFERENCE SYSTEM

The Multimedia Conference System has four layers: Client Layer, Load Balancer Layer, Server Cluster Layer, and Media Server Layer. The architecture is shown in Fig. 2.

A. Client

The main function of the Client is to send user requests, receive the server responses and present the login interface and main interface of the system. When the client receives an input event from the user, it sends the request to the backend server and then waits for responses. After receiving the response, the Client obtains the media data and other relevant data and shows them to the user.

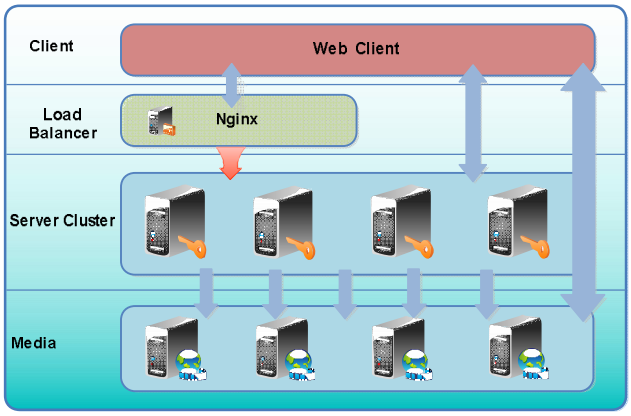

Figure 2. The Architecture of Multimedia Conference System

B. Load Balancer

The main function of the Load Balancer is to receive the request from the Client and verify if the request is legal. If the request is illegal, the page will be returned to the login interface and prompt error messages. If the request is legal, the load balancer will redirect the request to one of the application servers in the cluster according to the load balance strategy.

C. Server Cluster

The main function of the application sever in the server cluster is to receive and process the request from the upper layer and respond to the request. When the application server received a request, it will process it according to the type of the request. There are two types of the request. One is request for creating a conference and another is request for joining an existing conference. If the request is for creating a conference, the application server will make the user in the creating conference process. If the request is for joining a conference, the application server will make the user in the conference process.

D. Media Server

The main function of the Media Server is to collecting the audio and video and also including codec. The media server is controlled by the application server which can produce SIP signals send to the media server. Thus, the media server can provide the media resources that the Multimedia Conference System needs on the IP network.

IV. IMPLEMENTATION OF LOAD BALANCER

A. Framework of the Load Balancer

The framework of the Load Balancer Layer of the Multimedia Conference System is shown in Fig. 3.

The whole load balancer is divided into four modules: User Authentication, Decision-Making, Redirecting and Log Parsing. We will make a detailed introduction of these four layers in the following.

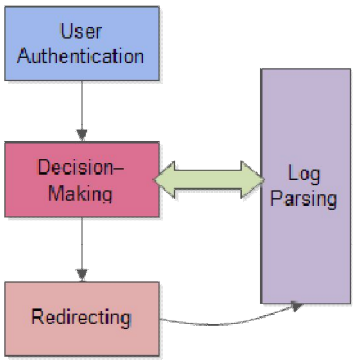

Figure 3. The Framework of the Load Balancer of Multimedia Conference System

- User Authentication

The User Authentication module is used for receiving the authentication information, including user name, user password and check code. This module will verify whether the user is legal according to the information. If the user is illegal, the page will turn back to the login interface and prompt with error message. If the user is legal, then the system will proceed according to the user's other choice.

- Decision-Making

If the user's authentication is successful, then enter the Decision-Making module. There are two identities in this system and the user can choose in the Client interface. Those who choose the button 'create a conference' will be the chairman, and those who choose the button 'join a conference' will be a participant. This module is used to distinguish the user's identity according to the choice that the user chose in the login interface. And the Decision-Making will make corresponding decisions according to their identities. If the user is a chairman, the process will enter the load balancing Redirecting module.

If the user is a chairman, the process will put the request to the Redirecting module directly. If the user is a participant, the process will get the real server IP from the database. The real server is a server in the Server Cluster on which the conference the user wants to join is held. And the real server IP will be an important parameter in the request which will be passed to the redirecting module.

- Redirecting

The Redirecting module is the http module in Nginx which is mentioned above in Fig. 1. The Redirecting module redirects the request from the Decision-Making module.

If the user is a chairman, the request will be redirected into the upstream resource pool. The instruction 'upstream' is mentioned in Fig. 1 and its main function is to implement load balance. Then the Nginx will choose an application server from the resource pool according to the load balance strategy [5] [6], such as Round-Robin algorithm and the IP-Hash algorithm. The application server which is chosen will process the request. At the same time, the log will record the corresponding relationship between the client IP and the real server IP which really processed the request. If the user is a participant, Nginx will redirect the request to the corresponding server according to the server IP which is contained in the request from the Decision-Making module.

- Log Parsing

The Log Parsing module is used to record the results of the Redirecting module, such as the corresponding relationship between the client IP and the real server IP.

If the user is a chairman who wants to create a conference, the Redirecting module will process user requests with a load balance strategy. After the request is redirected to one of the application servers, the log will record the redirect result. If the user is a participant who wants to join the conference, it will be necessary to get the real server IP on which the conference is held. The Log Parsing module can record the real server IP.

B. Key Technologies

1) Log Parsing

We can get the real server IP through Log Parsing. The real server is the one who really processed the user request. And the real server IP will be an important parameter for the subsequent participant who wants to join this conference.

General parameters used in the log are as the following Table I:

TABLE I. GENERAL PARAMETERS

Parameter	Explanation
$remote_addr	record the client IP
$remote_user	record the client user name
$time_local	record the access time and time zone
$request	Record the request URL and HTTP protocol
$status	record the request status, success is 200
$body_bytes_sent	record the body size send to the client
$http_referer	record the link from which access this page
$http_user_agent	record the client browser information

The parameters and '$'are used in the 'log_format' instruction.

In order to get the real server IP through the upstream module, there are three important parameters listed in Table II:

TABLE II. IMPORTANT PARAMETERS

Parameter	Explanation
$upstream_response_time	record the backend processing time and the agent's response time
$upstream_status	record the status of the backend
$upstream_addr	record the IP address and port of the backend

2) HttpUpstreamModule

The upstream module [7] is mainly used to do the load balance using the server resources which are listed in the upstream field. The algorithms used in this part including Round-Robin, the least number of connections and IP-Hash.

The upstream module has two main commands: upstream and server. In the upstream field, there are load balancing resources. In the server field, there is a location field. And in the location field, we can use "proxy_pass" to proxy a URL or use "rewrite" to rewritten the URL. There is an example shown in Fig. 4:

```
upstream backend  {
        server backend1.example.com weight=5;
        server backend2.example.com:8080;
        server unix:/tmp/backend3;
        }
server {
        location / {
        proxy_pass  http://backend;
                }
        }
```

Figure 4. An Example of Upstream Module

3) HttpRewriteModule

We can use the HttpRewriteModule (one of the Nginx modules) redirect the request that Nginx received. This module makes it possible to change URL using regular expressions (PCRE), and to redirect and select configuration depending on variables.

If the directives of this module are given at the server level, then they are carried out before the location of the request is determined. If in that selected location there are further rewrite directives, then they also are carried out. If the URI changed as a result of the execution of directives inside location, then location is again determined for the new URI.

Nginx rewrite rules related instructions are if, rewrite, set, return, break and so on. The 'rewrite' instruction is most important.

Syntax format for rewrite is:

rewrite *regex replacement [flag]*

Take the participant join a conference for an example, the redirect steps as follows:

- Get the real server IP that the participant wants to join;
- Match the user request with the location field in nginx.conf file;
- Using the PCRE (regex) in the rewrite field to match the server IP in order to redirect the request.
- The process of redirecting is over. Jump out of the rewrite module with the break instruction.

V. EXPERIMENTS AND EVALUATION

We have already installed [8] [9] the Nginx on the load balancer server with the Ubuntu operating system. The same Multimedia Conference System is deployed on every server in the server cluster. In order to see the effect of the load balancing, we marked the main interface of the Multimedia Conference System on every server with different markers to show the difference between servers. Now we will test the system in two aspects: the main interface with different marks and the logs that Nginx exported.

A. Experiment of Redirecting

The main purpose of this part is to verify whether the user can access the main interface of the system after redirecting.

The followings are detailed steps of the experiment:

Step 1: Input the user name, user password and check code for user authentication. The server will call the User Authentication module to process the authentication request. When the authentication is successful, the two buttons "create a conference" and "join a conference" can be pressed.

Step 2: Click "create a conference" button, a dialog box is shown, fill in the conference information and submit the form. After the background processing, the main interface is shown in Fig. 5.

Step 3: Repeat the above steps, and a main interface with a different marker is shown in Fig. 6.

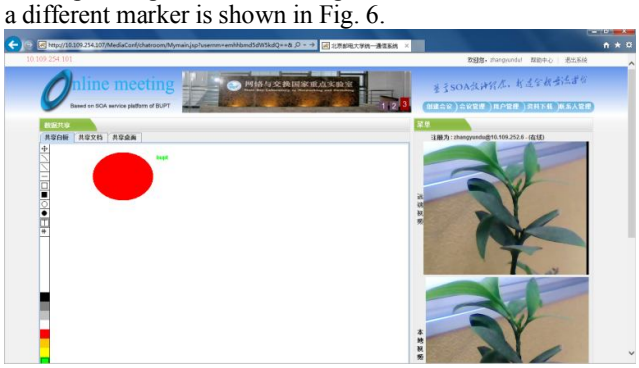

Figure 5. The main interface of the Multimedia Conference System

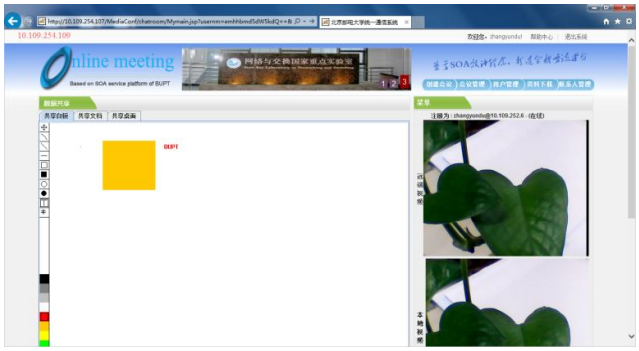

Figure 6. Another main interface of the Multimedia Conference System

From Fig. 5 and Fig. 6, we can see that the two interfaces have the same URL but different markers on the top of the main interface. Thus we know that after the process of the load balancer, the user's request was redirected to two different servers and the Redirecting module worked correctly.

B. *Experiment of Log Parsing*

The main purpose of this part is to check the real server IP and verify the process of load balance. Following is the log format in the nginx.conf.

```
log_format lblog  '$remote_addr'
                  '$remote_user [$time_local]  '
                  '$request" $status $bytes_sent '
                  '$http_referer $http_user_agent '
                  '$request_time $upstream_response_time '
                  '$upstream_addr $upstream_status ';
```

Figure 7. Log Format in the nginx.conf

After the user login and redirected, we can get the output of the log in Fig. 8 and Fig. 9.

```
8  10.108.165.146 - - [28/May/2013:09:44:34 +0800] "GET /MediaConf/ HTTP/1.1" 200 6108 "-"
   "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.95
   Safari/537.11" 0.003 10.109.254.107:8888 200
9  10.108.165.146 - - [28/May/2013:09:44:35 +0800] "GET /MediaConf/css/newLogin.css HTTP/1.1"
   200 4327 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11"  0.004 10.109.254.107:8888 200
10 10.108.165.146  - - [28/May/2013:09:44:35 +0800] "GET /MediaConf/js/ext-jquery-adapter.js
   HTTP/1.1" 200 15744 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1)
   AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.003
   10.109.254.107:8888 200
11 10.108.165.146  - - [28/May/2013:09:44:35 +0800] "GET /MediaConf/js/media.js HTTP/1.1" 200
   10142 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.003 10.109.254.107:8888 200
12 10.108.165.146  - - [28/May/2013:09:44:35 +0800] "GET /MediaConf/js/ext-all.js HTTP/1.1" 200
   4323"http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.002 10.109.254.107:8888 200
13 10.108.165.146  - - [28/May/2013:09:44:35 +0800] "GET /MediaConf/css/ext-all.css HTTP/1.1"
   200 5781 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.003 10.109.254.107:8888 200
14 10.108.165.146  - - [28/May/2013:09:44:36 +0800] "GET /MediaConf/checkCode.do HTTP/1.1" 200
   1100 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.001 10.109.254.107:8888 200
15 10.108.165.146  - - [28/May/2013:09:44:38 +0800] "POST /MediaConf/userLogin.do HTTP/1.1" 200
   86 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.002 10.109.254.107:8888 200
16 10.108.165.146  - - [28/May/2013:09:45:02 +0800] "POST
   MediaConf/chatroom/Mymain.jsp?usernm=emhhbmd5dW5kdQ==&passwd=MTExMTEx&userID=113 HTTP/1.1"
   200 10186 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.003 10.109.254.101:8888 200
```

Figure 8. Log of Nginx Recording Redirecting Information of the First Login

```
41 10.108.165.146  - - [28/May/2013:09:56:23 +0800] "GET /MediaConf/css/ext-all.css HTTP/1.1"
   200 5781 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.003 10.109.254.107:8888 200
42 10.108.165.146  - - [28/May/2013:09:56:23 +0800] "GET /MediaConf/checkCode.do HTTP/1.1" 200
   1100 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.001 10.109.254.107:8888 200
43 10.108.165.146  - - [28/May/2013:09:56:24 +0800] "POST /MediaConf/userLogin.do HTTP/1.1" 200
   86 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.002 10.109.254.107:8888 200
44 10.108.165.146  - - [28/May/2013:09:56:25 +0800] "POST
   MediaConf/chatroom/Mymain.jsp?usernm=emhhbmd5dW5kdQ==&passwd=MTExMTEx&userID=113 HTTP/1.1"
   200 10186 "http://10.109.254.107/MediaConf/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11
   (KHTML, like Gecko) Chrome/23.0.1271.95 Safari/537.11" 0.003 10.109.254.109:8888 200
```

Figure 9. Log of Nginx Recording Redirecting Information of the Second Login

The Fig. 8 shows that the login interface and user authentication is processed by the server whose IP is 10.109.254.107. The client get login interface element and push check code information to the 10.109.254.107. However, the main interface is processed by the server of 10.109.254.101.

The Fig. 9 shows that the login interface and user authentication is processed by the server whose IP is 10.109.254.107. The client get login interface element and push check code information to the 10.109.254.107. However, the main interface is processed by the server of 10.109.254.109.

Thus we can see that the load balancer processed the user authentication while the application servers in the cluster processed the other functions. This is consistent with previous framework mentioned in Fig. 2. And we also verified that the load balancer worked correctly.

VI. CONCLUSION

The Multimedia Conference System has made a great success and it provides a quick and convenient communication way, but a single server can't handle the high concurrent user requests effectively. In this paper, we improved the Multimedia Conference System with load balance technology. We have verified that with the load balance technology, the robustness and availability of the system can be enhanced significantly. But as an entrance of the cluster system, the load balancer may become another bottleneck. In the future work, we will put a focus on the entrance of the cluster system.

ACKNOWLEDGMENT

The work presented in this paper is supported by the National Grand Fundamental Research 973 Program of China (2011CB302506), the National Key Technology Research and Development Program of China "Research on the mobile community cultural service aggregation supporting technology" (2012BAH94F02), and the Novel Mobile Service Control Network Architecture and Key Technologies (2010ZX03004-001-01). We thank Wang Shaofeng, Weng Meizhen, Yang Nan and Lei Xiaojiang for their careful review and good suggestions.

REFERENCES

[1] Zhang Wensong, "The Study and Implementation of Scalable Network Services," Oct 2000.

[2] The Apache Tomcat Connector Documentation Index[EB/OL]. http://tomcat.apache.org /.

[3] nginx[EB/OL]. http://nginx.org/en/, 2013.

[4] Reese, Will. "Nginx: the high-performance web server and reverse proxy."Linux Journal 2008.173 (2008): 2.

[5] Luo Yongjun,Li Xiaole and Sun Ruxiang, "Summarization of the Load-balancing Algorithm," Nov 6,2008.

[6] Keslassy, Isaac, et al. "Optimal load-balancing." INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. Vol. 3. IEEE, 2005.

[7] Tian Chunqing, "Using Nginx to Implement Web Load Distribution Based on URI,"April 16,2009.

[8] Deng Zhongju, "The Design and Implementation of High Reliable Cluster's Deploymen,"Dec 25,2012.

[9] Chi, Xiaoni, et al. "Web Load Balance and Cache Optimization Design Based Nginx under High-Concurrency Environment." Digital Manufacturing and Automation (ICDMA), 2012 Third International Conference on. IEEE, 2012.