Highly Available and Efficient Load Cluster Management System using SNMP and Web

Myung-Sup Kim, Mi-Jeong Choi and James W. Hong
Dept. of Computer Science and Engineering
POSTECH
Pohang, Korea
{mount, mjchoi, jwkhong}@postech.ac.kr

Abstract

To cope with the explosive increase in the number of requests to Internet server systems, one popular solution is a load-balancing technique that uses a dispatcher in the front-end of a cluster farm. A cluster group is viewed as a single system image with very high performance and also gives a good scalability. But a failure in any single host in a cluster group can cause an overall system failure. The high availability in a cluster group is desperately needed for stable and fault-tolerant service to clients. So it is necessary to develop a cluster management system that integrates all these cluster functions and user-friendly management functions.

In this paper, we present the design and implementation of a load cluster management system (LCMS) based on SNMP and Web technology. Our LCMS implementation has been deployed on a commercial ultra dense server like an EnterFLEX. First we examine the requirements of LCMS to provide efficient and stable management operations and high availability. Our LCMS follows the client-server management paradigm of SNMP, and consists of three managers having different roles, which distribute management functionality to all hosts in a cluster group. By using SNMP we can reduce the network bandwidth required in management operations. This system also provides automatic cluster configuration and current status monitoring of each host in a cluster group through a Java and Web technologies.

Keywords

Load Cluster Management System, Load Balancing, System Management, SNMP, Web-based Management.

1. Introduction

An Internet server farm [1, 2] using load balancing is one solution to the enormous growth of requests to Internet server systems. This solution will have wide-ranging ramifications for the set up of high-performance Internet service sites

because of several attractive features. It is a very cost-effective solution. We can compose a load cluster system using off-the-shelf and low-price computers. It also provides good scalability and extendibility, so we can easily add a single server into a load cluster group to increase service throughput, and remove a single point of failure during its operation. From a client viewpoint, a load cluster group is considered to be a single powerful system with a single hostname and IP address. It is not necessary to change in the client side to obtain service from a load cluster group.

There are several implementation methods to construct an Internet load cluster system, which includes a round-robin Domain Name Server (RR-DNS) approach [2, 3, 4], a parallel filtering approach [7] and a dispatcher approach [10, 11]. But RR-DNS and parallel filtering cannot consider the current exact workload of real servers in the cluster farm for a real server selection algorithm. Other problems arise, such as the difficulty in detecting a single server fault and recovery. Further, in the RR-DNS approach, the catching in local DNS has a locality problem [5] where all requests from hosts at a single domain are forwarded to a single server, so it can be easily overloaded. In the dispatcher approach, all requests from the clients are caught by a single host and are forwarded to the appropriate host. The best examples of this type are the L4 switch [6, 7] and the Linux Virtual Server (LVS) [8, 9, 10, 11]. This type of load cluster system is widely used because of its good performance and throughput.

To operate this kind of load cluster system, supporting a fault tolerant and stable service is very important. The efficient use of resources is also a key point to be considered in the design of a load cluster management system. A cluster server is a set of low performance machines such of PCs but this provides high performance as Internet server. The possibility of a single server failure is higher than one single server system. In the case of failure of a real server or a load balancer, a high available cluster management system reconfigures the cluster farm by removing this failed host in the service group or replacing the activity with another host, so the service to client continues. Much research on the load cluster management system focuses on automatic cluster configuration only, and resource management is not efficient, and the high availability (HA) functionality is separated from the cluster management system.

In this paper, we developed a load cluster management system that integrates all these cluster functions and user-friendly management functions. We first discuss the requirements of the load cluster management system and design a load cluster management system, which provides efficient resource management and good HA features. We have implemented our LCMS on a Linux platform using SNMP, Java and Web technologies. Our LCMS implementation has been tested on a commercial ultra dense server called Netstech EnterFLEX 2100 [12].

The organization of this paper is as follows. In Section 2, we present an overview of the load cluster system and discuss some related work on cluster management systems. In Section 3, we discuss the requirements of the load cluster management system. In Section 4, we present the design of load cluster management system supporting HA and good reliability in detail. In Section 5, we

present implementation issues. In Section 6, we summarize our work and discuss possible future work.

2. Related Work

In this section, we discuss some related work on cluster management system. In this paper we distinguish a load cluster system from a super cluster system. We define a load cluster system is a server farm where each real server performs an independent Internet service such like web and ftp services. The best example of a load cluster system is Linux Virtual Server (LVS) [8, 9, 10, 11]. And a super cluster system is a group of servers where each real host performs one part of a certain job which needs a very much computation time. At first, we present an example of a load cluster system that is one of network-level dispatcher approaches. And then we present some existing load cluster management systems and super cluster management systems. We also compare weak and strong points of these systems.

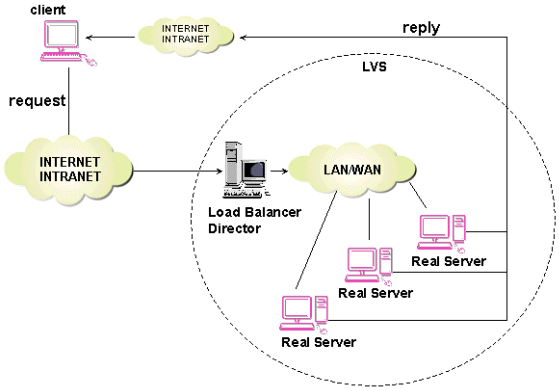

Figure 1: LVS Service Architecture

One example of a load cluster system using the dispatcher approach is the Linux Virtual Server (LVS) LVS consists of one load balancer and several real servers. The load balancer receives all requests from clients and distributes them to a real server using an appropriate scheduling algorithm, such as round-robin (RR), weighted round-robin (WRR), least-connection (LC) or weighted least-connection (WLC). Three types of IP load balancing techniques exist together in the load balancer. They are virtual server via NAT, IP tunneling and Direct Routing. The parallel services of real servers can be made to appear as a virtual server on a single IP address, so that the end users see a virtual server, not a cluster of servers. Scheduling granularity is per connection, which can create a sound load balance among the servers. This type of IP-level load balancing is better than application-level load balancing, such as reverse-proxy [13]. Figure 1 illustrates an example of LVS service architecture.

Now we discuss some of the most popular load cluster management systems that are currently available in the market: Turbo Cluster Server 6 [14] from TurboLinux Inc. and RedHat HA Server (Piranha) [15] from RedHat Inc..

Turbo Cluster Server 6 provides a good configuration tool and has powerful and well-categorized menu system. It also provides a useful tool: a data synchronization module between real servers. Further, it provides a simple web-based configuration and status display tools. However the configuration menu is consol-based, so a new user cannot easily learn how to operate it. The alarm notification functionality is not provided in this cluster management system.

Using Piranha, we can construct two types of cluster systems: Fail-Over Service (FOS) and Linux Virtual Server (LVS). It provides Web and Windows based configuration tools. But these tools are dedicated only to configuration functionality and provide a primitive monitoring and status reporting menus. The current version of Piranha does not support alarm notification and log files.

Turbo Cluster Server 6 and Piranha are a distributed configuration system. Both are very weak in status monitoring and lack a reporting mechanism. Other load cluster management system like HP WebQoS[16], IBM WebSphere Edge Server[17] and CISCO LoacalDirector[18] is also good commercial management system which give much of management functions and high availability functions. But these systems have a dedicated dispatcher and backup server which cause system redundancy.

Table 1: comparison of cluster management system

	CEO	JobCoNTrol	NCSA Symera	Turbo Cluster	Piranha
Management Type	Distributed	Distributed	Distributed	Centralized	Centralized
Communication Method	Proprietary	Proprietary	DCOM	Proprietary	Proprietary
Manager Interface	GUI	GUI	GUI	Consol/Web	GUI/Web
Managed target	UNIX/ Windows	Windows NT	Windows	Linux Windows	Linux Windows
Cluster Type	HPC	HPC	HPC	LBC	LBC
Cluster Configuration	X	X	X	O	O
Fault Tolerant	X	X	O	O	O
Resource management	O	O	O	X	X
Portability	O	X	X	X	X

O: supported, X : not supported

Now, we discuss a super cluster management system (CMS). Generally, CMS is an interface to a cluster of computers, which is used for executing parallel programs in a high performance cluster system using a parallel processing library such as MPI [19]. Until now, we could not locate any integrated management system for the load cluster system, such as CMS on high performance cluster. Our LCMS is based on CMS in the basic idea and modified the functionality and management model.

Recently, many CMSs have been proposed and developed. Most CMS architectures are based on the manager/agent paradigm [20, 21] and use his own communication protocol between a manager and an agent. A specific CMS [21, 22] is developed to manage a specialized target, such as an NT cluster, but this specialized system causes a portability problem. The main functions of CMS are

status monitoring and job assignments, which do not consider fail-over in the case of a single system failure. GUI-based user interfaces are typically implemented on a Windows application, so the administrator can access and control a cluster system using only a specified computer on which CMS is running. The existing CMS for load balancing mainly focuses on HA functionality to provide a stable service. Therefore, cluster configuration and status monitoring was not considered. Table 1 illustrates the comparison of several super and load cluster management systems. The load cluster management system (LCMS) presented in this paper is an integrated system with all functionalities to operate a load cluster system. LCMS should provide three major functionalities: load cluster configuration, high availability, current and past cluster system status monitoring. Next, we consider these functional requirements for LCMS in more detail.

3. Requirements of LCMS

Before considering LCMS design, we should consider the functional requirements of LCMS. We offer this discussion as a guideline for further LCMS designs. These requirements should be applicable to all types of load cluster systems and should not be biased towards a certain implementation method. We categorize LCMS requirements into six types:

•**Efficient Resource Management:** This requirement concerns host management and cluster group management. In host management, LCMS should be able to store the information about hosts that are members of a load cluster group. The information can include the host name, the IP address, CPU type, memory size, etc. When a new cluster group or service group is created, this information should be stored. The LCMS should manage all this information and provide a method to lookup or update host and cluster information to administrators.
•**Load Cluster Configuration:** A cluster group consists of two or more hosts where one host acts as a load balancer and the others as real servers. Further, there can be a dedicated backup server in a cluster group.
LCMS should provide a way to assign a role to each host when a new load cluster group is created or an already existing cluster group is modified or deleted. And an administrator should be able to check the current status of each host and cluster group.
•**High Availability:** This can be the main functionality of LCMS for a fault-tolerant service. LCMS should check the current state of each host in a cluster group. If a host in a cluster group fails it should recover by load cluster reconfiguration. Reconfiguration can be executed by removing or replacing a failed host with a new one.
• **Effective management Interface:** LCMS should provide GUI-based cluster configuration and a visualized status monitoring method for management convenience.
•**Security:** All administrator, host and cluster information should be stored safely and a user authentication method should be provided. Only an authorized

administrator should access the management system. Another security issue on LCMS is that the communication between manager component and agent component should be secured.

• **Minimization of Management Overhead:** Management functionality is important but is not the main function of a load cluster system. Therefore, LCMS should not create high network overhead. Management overheads should be distributed among hosts in the cluster group so as not to overburden a specific host.

4. System Design of LCMS

Figure 2 is an overall architecture of our Load Cluster Management System (LCMS), which is designed to satisfy the requirements we discussed in the previous section. Our LCMS consists of three kinds of manager: Load Cluster (LC) Manager, Load Balancer (LB) Manager and Real Server (RS) Manager. These three managers have different functionalities and are distributed among hosts in the cluster. This LCMS is designed for the dispatcher types of load cluster.

The LC manager is responsible for configuration. To store host and cluster information, an information repository is used. Administrator information is also stored in this repository. An administrator can add and remove a host into the information repository through the web interface provided by the LC manager and can make a new load cluster group and modify or delete an existing load cluster group. The LB manager is responsible for status monitoring of each cluster group. It periodically checks the status of the load balancer in each cluster group. When a load balancer is down or does not work well for some reason, the LB manager chooses one among all real servers in the cluster group and set this real server as a load balancer. This real server works as the load balancer until the original load balancer wakes up and plays its role again.

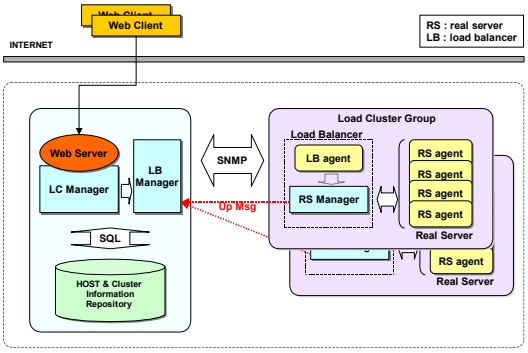

Figure 2: Load Cluster Management System

When an administrator creates a new cluster group, he must select one host as a load balancer, and the other hosts run as real servers. The RS manager is running at the load balancer in each cluster group. It is responsible for the HA functionality.

The RS manager periodically checks whether each real server plays his role well or not. If a real server fails, the RS manager removes this host from the request-forwarding list in a load balancer and includes this host again when this host is running again.

All hosts that belong to a load cluster group have an SNMP agent [23]. There are two types of SNMP agents: an RS agent and an LB agent. Between a manager and an agent, the communication is performed via SNMP over UDP, so LCMS reduces network traffic for management functionality.

The manager interface is composed using Web technologies such as JAVA and HTTP, so the administrator can easily access the LCMS from any computer connected to the Internet and perform all management functions, such as displaying historical data, as well as the current status of each host and cluster.

4.1. LC/LB Manager Architecture

Cluster configuration and resource management are the main roles of the LC manager, which is illustrated in Figure 3. The administrator and host are resources to be managed by the LC manager. The information of these resources is stored in the Information Repository (IR). The LC manager provides a web-based UI for user authentication, resource management, and cluster configuration.

As an Information Repository the LC manager uses a database. In the following subsection, we discuss in detail how host and cluster information is stored in the IR. The LC manager and DB server can be located in different hosts, which gives workload distribution.

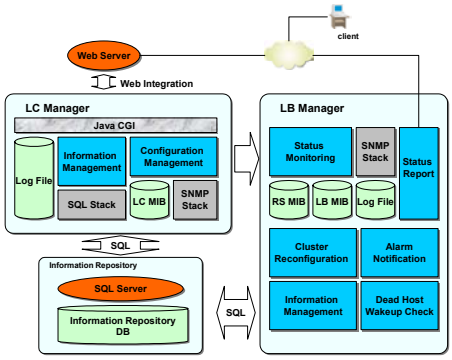

Figure 3: LC and LB Manager Architecture

There are three main components in the LC manager: a status monitor, an information manager, and a configuration manager. The configuration manager is responsible for the creation and deletion of a load cluster group, and uses SNMP to communicate with each host and SQL to store cluster information. The cluster configuration sequence is as follows. First, the administrator views the host list stored in the IR with the assistance of the information manager. Next, the administrator selects a set of hosts and selects one host as a load balancer to create

a new cluster group. Next, the administrator can create one or more service groups in that cluster group. A host cannot belong to more than two cluster groups. Figure 4 illustrates the cluster and service assignment to hosts in the IR.

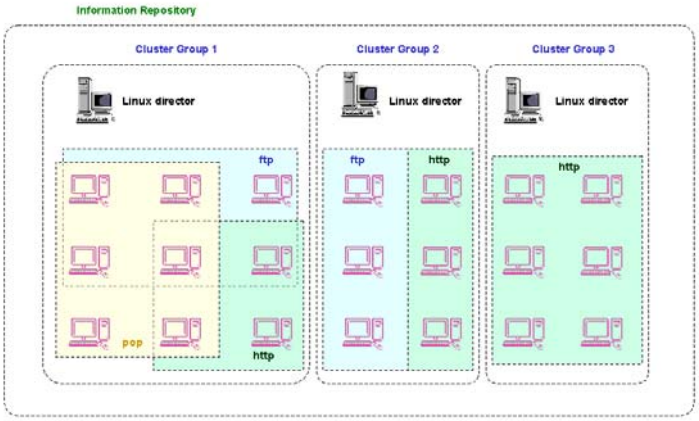

Figure 4: Cluster and Service Group Construction

Role assignment to each host is made by SNMP. The agent receives SNMP packets and sets the system to begin its service. All actions by the administrator are recorded into a log file and also are viewed in a web client.

The role of the status monitor in the LB manager is to check the status of a load balancer in each cluster group and take appropriate action when one is down or malfunctions. All significant actions are recorded in a log file and load balancer malfunctions are reported to the administrator by e-mail.

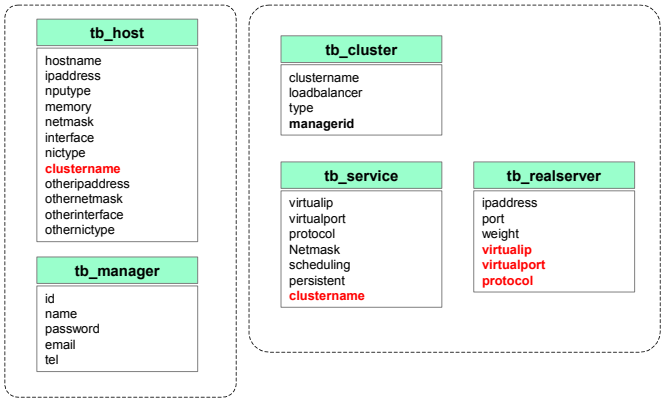

Figure 5: Tables in Information Repository

4.2. Information Repository Structure

To store and maintain host and cluster information we used the DB as an Information Repository. The data stored in the LCMS information repository can be categorized into three groups. One concerns administrator information, the second concerns host information and the last concerns cluster group information.

The tables in Figure 5 indicate the attributes of each information group. Host information has several attributes, such as host name, CPU type, memory size, etc. When the administrator adds a new host to an LCMS, the host information is gathered from the new-added host by SNMP, and stored at the DB table. Because one load cluster group can have several service groups and one service group can have several real servers, there are three tables to indicate cluster information

4.3. RS Manager Architecture

The RS manager illustrated in Figure 6 is running at a load balancer in a load cluster group, which is executed when a cluster group is defined and an administrator decides a host as a load balancer. To accomplish this, the LC manager sends a SNMP request message to set the host as a load balancer.

The functions of the RS manager is to monitor the up-to-date status of all real servers in which the service daemon, such as a web server or a streaming server, is running to provide its services to the client. If a real server fails, it removes the real server in its load cluster group and reconfigures the load balancer. This periodic check of the real server state is performed using the SNMP protocol. The SNMP agent in a real server has a module to check the service state of the real server. By this SNMP communication for monitoring the real server, LCMS can reduce network bandwidth overhead and distribute the management workload to hosts in the load cluster group. Another advantage of this manager/agent architecture is that the manager can obtain the exact up-to-date workload of all real servers, such as CPU usage, memory usage, the number of current service connections, and outgoing network bandwidth. This information can be used for the request packet-scheduling job. To monitor all real servers, the RS manager uses the RS MIB.

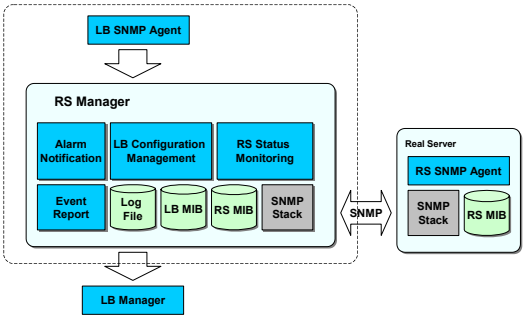

Figure 6: RS Manager Architecture

From the time the RS manager is executed, the status of all real servers is periodically monitored and the status data is stored in log files. When a real server

fails, the alarm module executes by sending e-mail to the administrator and sending a message to the LB manager.

4.4. LCMS Agent

In the LCMS architecture, two types of agents perform different tasks. This corresponds to two types of managers: an RS agent and an LB agent. If a host acts as a load balancer, the LB agent runs on this host. If a host runs as a real server, the RS agent is executed. Every host has the potential to be a load balancer and a real server, and all agents should be able to run in all hosts. The role of each host in a load cluster group is assigned by the manager through an SNMP message. The agent should be able to set its local host as a load balancer and a real server.

From the consideration of these special features in a load cluster system, the LCMS MIB is defined. Figure 7 shows the LB MIB in the LCMS MIB. When the LC manager assigns a role to a host, the agent sets the *agentType* field in the *generalInfo* category with the value of 0, 1, 2 and 3. If the *agentType* field is set to 1, the OIDs under the load balance are only meaningful and the host is running as a load balancer. In the same manner, if a host is running as a real server, the OIDs under the real server is meaningful, the backup server and the value of *agentType* is 2.

In one cluster group, one or more services can be created, as in Figure 4. The load balancer in a cluster group can provide more than one service, such as web service, FTP service, and streaming service. Therefore, the real server group can differ from each service. We defined two MIB tables to store service information and real service information in a single load balancer. By the attribute *rsSrIndex* in the *realServerTable* the relationship between real server and service can be found, which is illustrated in Figure 7.

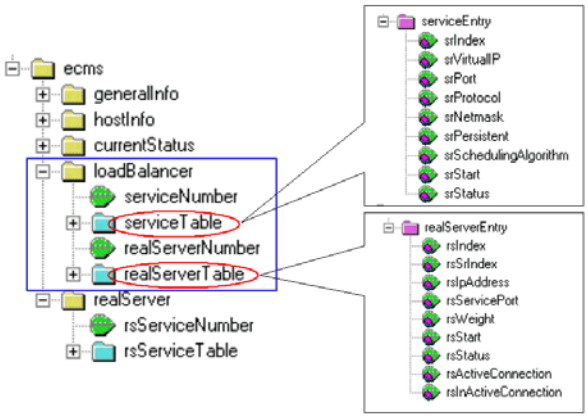

Figure 7: LB MIB in LCMS MIB

In the current version of LCMS MIB, the security check between the manager and the agent is performed by SNMP v1 community name [23]. This is intended to

be replaced to the user-based security model in SNMPv3 [24] in our next version of LCMS.

4.5. High Availability and Resource Management

The RS manager and the LB manager are responsible for processing faults in the cluster group. The RS manager takes action when a real server fails to perform its assigned service, as presented in the previous section. Next, we discuss in more detail how the RS and the LB manager operate when a fault occurs in a host.

First, when a real server has a fault, the RS manager detects it and removes this host among a request-forwarding list in a load balancer, and report to the LB manager. The LB manager updates IR that this host is down. Afterwards, the RS manager and the LB manager periodically check this host until this host wakes up again. When this dead host wakes up, the RS manager reports to the LB manager. Then the LB manager reads the role assigned to this host from IR and send SNMP message for the job assignment. Also, the LB manager can detect that this host wakes up again. If the RS manager sent a message already, then LB manager does nothing. But if the RS manager has not sent a message yet, the LB manager performs the job assignment action to this host. In this way, we increased the HA by two managers checking the status of a dead real server.

Second, when a load balancer is dead, the LB manager selects a host that does the least services and has the smallest workload among all real servers in the cluster group. Then the LB manager assigns the role of a load balancer to that real server. During that time the real server works as a load balancer not as real server. When the original real server wakes up again, the role is assigned back to the original. Among a cluster group, there is no redundant backup server. We increased the efficiency of the resource usage. In our cluster composition architecture, there is no dedicated backup server for a load balancer. That means any real server can function as a backup server when a load balancer fault occurs.

5. LCMS Implementation

Using this design architecture, we have developed LCMS, which is deployed on a commercial ultra dense server called Netstech EnterFLEX 2100 [12]. The software is named EnterFLEX Cluster Manager. The current status of development is that we almost finished the first version and are doing some beta testing.

For LCMS, we used MySQL DB [25] as an information repository, which runs in the same machine with the LC manager. The UCD-SNMP and its extension are used to make an LB and RS agent at each host. The SNMP agent is programmed using C language. All managers and user interfaces are developed with Java technologies such as JDK, JSP, Servlet and Applet. We used the joeSNMP library [26] provided by OpenNMS for SNMP communication between manager and agent. The LC manager is made usually in the form of JSP, and RS manager and LB manager is made of a Java application.

We used JSP, Applet, and Servlet for the user interface. Tomcat 3.2.1 [27] is used as a JSP engine, and Apache web server is used as front-end web server. We have selected Java/JSP for LCMS because it is flexible, portable, easy to develop, and has a rich class APIs.

Figure 8: LCMS & Streaming Service Pages

To validate our LCMS, we first developed a streaming service cluster system using a Linux Virtual Server. The streaming service on the Internet is performed by two server types. One is by a web server and the other is a specialized streaming server. The Windows Media Player from Microsoft[28], Real Server from Real Networks[29] and Quicktime Server from Apple[30] are generally used in the current Internet environment. We set up a streaming service cluster supporting these three kinds of media formats and two types of servers in the Linux environment. We used a direct routing method of LVS load balancing. We used an NFS file server to store all stream media files, which is shared among all real servers. The Apache server and the streaming server are used on the real server side. Streaming service web page has been developed using JSP in JAVA.

The screen shots in Figure 8 show selected user interfaces of the streaming service and LCMS, and the screen shots in the bottom are LCMS user interfaces for information repository operation and new cluster generation.

6. Conclusion And Future Work

The load cluster system is a cost-effective solution to construct a high-performance Internet server system. As deployment ratios of the load cluster system increase year after year, we need a good management solution. In this paper, we proposed a new management method for a load cluster system, which is based on SNMP, the Web and DB.

The contributions of this paper are as follows. The first contribution is management workload distribution. In this type of dispatcher method, a load

balancer can be easily overloaded and can be a single point of failure. Our proposed LCMS reduces the HA workload in a load balancer by separating HA functionality into an RS manger and an RS agent. The second contribution is that LCMS minimizes the network bandwidth needed for management functions using SNMP. Our last contribution is to efficiently manage the host and load cluster information using an information repository. LCMS follows the current trend of web-based user interface for manager convenience.

Our future work is as follows. It is necessary to measure the exact workload of our LCMS, and compare it with other load cluster management systems. This LCMS is designed and developed over the dispatcher approach of load balancing, so we plan to apply this LCMS to other types of load balancing technique. We also plan to apply the management information into a request packet-scheduling algorithm. We anticipate that the load balancer will use this information as CPU load, memory usage, and current network bandwidth, that a client requests and service workload will be more equally distributed among real servers.

References

[1] Jonathan Cragle, "Load Balancing Web Servers", Windows2000 Magazine, June 1998.

[2] T. Brisco, "DNS support for load balancing", RFC 1794, April 1995, http://www.ietf.org/rfc/rfc1794.txt.

[3] T. T. Kwan, R. E. McGrath, and D. A. Reed, "NCSA's World Wide Web Server: Design and Performance", IEEE Computer, November 1995, vol. 28, no. 11, pp.68-74.

[4] E. D. Katz, E. D. Katz, R. McGrath, "A Scalable HTTP Server: The NCSA Prototype", Computer Networks and ISDN Systems, Nov. 1994, vol. 27, no. 2, pp.155-163.

[5] Michele Colajanni and Valeria Cardellini, Philip S. Yu, "Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers", Proceedings of the The 18th International Conference on Distributed Computing Systems, ICDCS 1998.

[6] Foundry Networks, "Cutting Through Layer 4 Hype", white paper, http://www.foundrynet.com/whitepaper_layer4.html.

[7] Mohit Aron, Darren Sanders, Peter Druschel, Willy Zwaenepoel, "Scalable Content-aware Request Distribution in Cluster-based Network Servers", USENIX Annual Technical Conference, June 2000.

[8] Tewari R, Dias D, Mukherjee R, Vin HM "High Availability for Clustered Multimedia Servers", Proceedings of International Conference on Data Engineering, February 1996, IEEE Press, pp.345-354.

[9] A. Robertson, "High-Availability Linux Project", May 1998 - now, http://www.linux-ha.org/.

[10] Wensong Zhang, "Linux Virtual Server Project", May 1998 - now, http://www.linuxvirtualserver.org.

[11] Wensong Zhang, "Linux Virtual Server for Scalable Network Services", Ottawa Linux Symposium 2000, July 2000.
[12] Netstech, EnterFLEX 2100 user guide 2.0, http://www.netstech.com, July 2001.
[13] Ralf S. Engelschall, "Load Balancing Your Web Site: Practical Approaches for Distributing HTTP Traffic ", WebTechniques, May 1998,http://www.webtechniques.com/archives/1998/05/engelschall/.
[14] TurboLinux, Turbo Linux Cluster Server 6 user guide, http://www.turbolinux.com, 2000.
[15] RedHatLinux, Piranha white paper, http://www.redhat.com/support/wpapers/piranha/index.html, 2001.
[16] HP WebQoS,HP, http://www.hp.com/products1/webqos/.
[17] IBM WebShephere, IBM, http://www-3.ibm.com/software/webservers/edgeserver/.
[18] CISCO LocalDirector, CISCO, http://www.cisco.com/univercd/cc/td/doc/product/iaabu/localdir/diruser/.
[19] MPI Forum, "The Message Passing Interface (MPI) Standard", http://www-unix.mcs.anl.gov/mpi/.
[20] Rajkumar Buyya, Toni Cortes, Oriol Teixio, "Cluster Environment Observer", Monash University, Australia, http://www.csse.monash.edu.au/~rajkumar/ClusterObserver/.
[21] Markus Fischer, "A cluster management software for Windows 9X, NT, 2000", 1999, http://mufasa.informatik.uni-mannheim.de/lsra/persons/markus/jobcontrol.htm.
[22] The Symera Team, "NCSA Symera", 1997-1998, http://symera.ncsa.uiuc.edu/.
[23] William Stallings, "SNMP, SNMPv2, and RMON 1 and 2", Addison Wesley, 1999, pp163-203.
[24] U. Blumenthal, B. Wijnen, , "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) ", April 1999, RFC2574, IEEE, http://www.ietf.org/rfc/rfc2574.txt.
[25] Net SNMP Project Team, "Net-SNMP", Soruceforge, http://net-snmp.sourceforge.net/.
[26] OpenNMS, "joeSNMP Library 0.2.5", http://www.opennms.org.
[27] Jakarta Project, "Tomcat", http://jakarta.apache.org/tomcat/index.html.
[28] Microsoft, "Microsft Windows Media", http://www.microsoft.com/windows/windowsmedia/en/default.asp.
[29] Real Networks, "Real media technology", http://www.realnetworks.com/.
[30] Apple, "QuickTime", http://www.apple.com/quicktime/.