Cluster-based DBMS Management Tool with High-Availability

Jae-Woo Chang, Young-Chang Kim
Dept. of Computer Engineering, Chonbuk National University
Chonju, Chonbuk 561-756, South Korea

ABSTRACT

A management tool which is needed for monitoring and managing cluster-based DBMSs has been little studied. So, we design and implement a cluster-based DBMS management tool with high-availability that monitors the status of nodes in a cluster system as well as the status of DBMS instances in a node. The tool enables users to recognize a single virtual system image and provides them with the status of all the nodes and resources in the system by using a graphic user interface (GUI). By using a load balancer, our management tool can increase the performance of a cluster-based DBMS as well as can overcome the limitation of the existing parallel DBMSs.

Keywords: Cluster management tool, cluster-based DBMS, high-availability

1. INTRODUCTION

Cluster systems developed by connecting PCs and workstations using high-speed network [1] is required to support 24-hours nonstop service for the Internet. Therefore, there are a wide range of researches on cluster-based DBMSs that offer a mechanism to support high performance, high availability, and high scalability [2,3,4]. They include Oracle 9i Real Application Server, Informix Extended Parallel Server and IBM DB2 Universal Database EEE. To manage the cluster-based DBMS efficiently, a management tool for the cluster-based DBMS is needed. First, the tool enables users to recognize a cluster system consisting of multiple nodes as a single virtual system image. Secondly, by using a graphic user interface (GUI), the cluster-based DBMS management tool provides users the status of all the nodes in a system and all the resources (i.e., CPU, memory, disk) in a node. Thirdly, a load balance function is needed to make all the nodes perform effectively by evenly distributing user's requests. Finally, a fail-over technique is needed to support high availability when the node failure is occurred [5,6].

In this paper, we design and implement a cluster-based DBMS management tool which monitors the status of all the nodes in a cluster system as well as the status of DBMS instances in a node. The tool enables users to recognize a single virtual system image and provides them with the status of all the nodes and resources in the system by using a graphic user interface (GUI). In addition, our cluster-based DBMS management tool with a load balancer can increase the performance of a cluster-based DBMS as well as can overcome the limitation of the existing parallel DBMSs.

The rest of this paper is organized as follows. The next section discusses related work on existing cluster management tools. In section 3, we design a cluster-based DBMS management tool and its graphic user interface. In section 4, we describe the implementation and the performance analysis of our cluster-based DBMS management tool. Finally, we draw our conclusions in section 5.

2. RELATED WORK

In this section, we introduce the existing management tools; the OCMS(Oracle Cluster Management System) [7] which is well known as a cluster-based DBMS management tool and the SCMS(SMILE Cluster Management System) [8] which is a cluster system management tool for Linux Beowulf. OCMS is included as a part of the Oracle8i Parallel Server product on Linux and provides cluster membership services, a global view of clusters, node monitoring, and cluster reconfiguration. It consists of the watchdog daemon, node monitor, and cluster manager. First, the watchdog daemon offers services to the cluster manager and to the node monitor. It makes use of the standard Linux watchdog timer to monitor selected system resources for preventing database corruption. Secondly, the node monitor passes node-level cluster information to the cluster manager. It maintains a consistent view of the cluster and informs the status of each local node of the cluster manager. The node monitors also cooperates with the watchdog daemon to stop the node with the abnormally heavy load. Finally, the cluster manager passes instance-level cluster information to the Oracle instance. It maintains the process-level status of a cluster system. The cluster manager accepts the registration of Oracle instances to the cluster system and provides a consistent view of the Oracle instances. Figure 1 shows the overall architecture of OCMS.

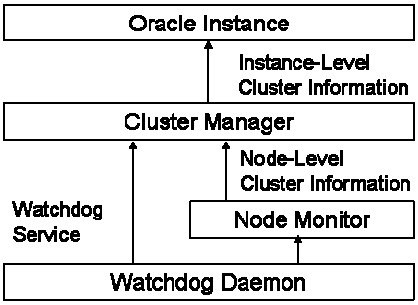

Figure 1. Overall architecture of OCMS

The SCMS is developed by the Kasetsart university in Tailand as a cluster system management tool for Beowulf cluster. It consists of CMA (Control and Monitoring Agent), SMA(Systems Management Agent), and RMI(Resource Management Interface). First, the CMA runs on each node and collects system statistics continuously. The CMA reads system information through a layer called HAL(Hardware Abstraction Layer). Secondly, the system statistics are collected by a centralize resource management server called SMA. The SMA responses a user query on a system status and sends some

commands to the CMA. Finally, the RMI is provided as a set of APIs for system monitoring and logging applications. By using the RMI, the SCMS provides monitoring softwares, configuration utilities, and parallel Unix commands. Figure 2 shows the overall architecture of SCMS.

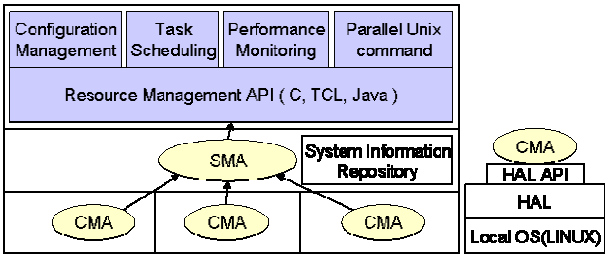

Figure 2. Overall architecture of SCMS

3. DESIGN OF CLUSTER-BASED DBMS MANAGEMENT TOOL

The cluster-based DBMS consists of multiple server nodes and uses a shared disk. Each server node is connected with each other by using high-speed gigabit Ethernet. A master node performs both roles of a gateway of the cluster system and a server node. A master node manages all the information gathered from all the server nodes and makes use of a Linux virtual server for its scheduling algorithms. First, a user service request is transmitted to the master node by using virtual IP. The Master node send it to a server node selected by the scheduling algorithm of the Linux virtual server. The selected server node processes the user request and returns a result to the user. Figure 3 shows the overall architecture of cluster-based DBMS using high-speed gigabit Ethernet.

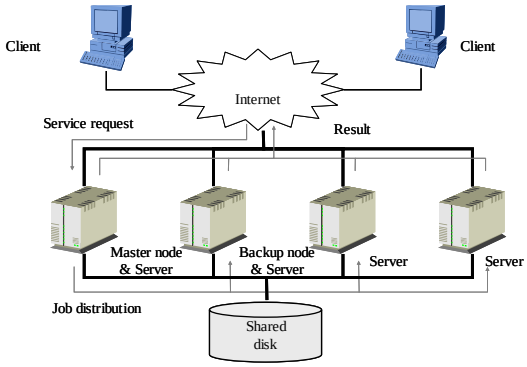

Figure 3. Overall architecture of Cluster-based DBMS

Cluster-based DBMS Management Tool

The cluster-based DBMS management tool monitors both the status of system resources and database instance in each node. It also perceives the error of each node and performs its recovery procedure to make a cluster-based DBMS run in a normal situation. To design a good monitoring tool, we first minimize the objects to be monitored so that we may avoid the additional load of monitoring itself. Secondly, we change the frequency of monitoring dynamically so that we may control the amount of network traffic where a node transmits its status information to the master node. The cluster-based DBMS management tool consists of four components; probe, handler, CM(Cluster Manager), and NM(Node Manager). In addition, the probe (or handler) can be classified into system, DB, and LB(Load Balancer) probes (or handler). Figure 4 shows the components of the cluster-based DBMS management tool.

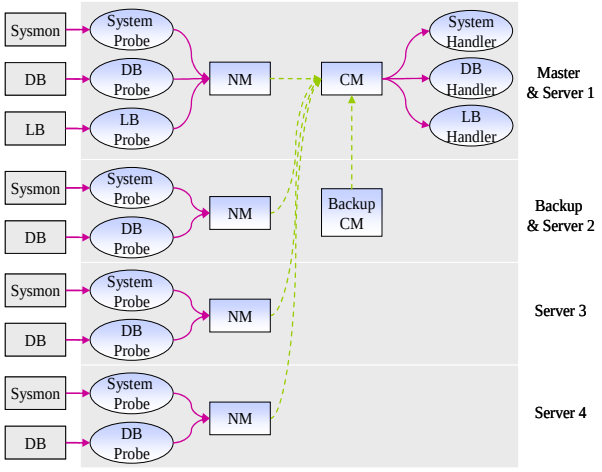

Figure 4. Components of cluster-based DBMS management tool

System Probe & System Handler: The system probe monitors the status of CPU, memory, disk, and network of each node. For this, we uses /proc virtual file system to gather the status information of the main system resources, such as CPU, memory, and disk, as well as the transmission and reception status of packets through the network. The system probe also generates events according to the status of the system. When the monitoring is performed without errors, the system probe sends the events and the monitored information to the system handler. The system handler stores the events and the monitored information into the service status table, and it performs a procedure according to the events. When an error occurred in the network, the system probe updates the service status table. If the system probe or NM has a failure, the system handler makes them restart.

DB Probe & DB Handler: In the DBMS side, the DB Probe monitors the usage rate of CPU and memory when DBMS is running and generates events as a status of DBMS. When the cluster-based DBMS runs without an error, the DB probe generates 'DB_ALIVE' event. The DB Handler maintains the service status table and performs a procedure according to the event. In case the cluster-based DBMS has a failure, the DB probe performs a procedure to recover the transaction, removes the failed server node from the available server list, and makes the cluster-based DBMS restart. It also performs a recovery procedure according to the network error perceived by CM.

LB Probe & LB Handler: The LB probe monitors a load balancer and generates event as a status of load balancer. For this, we use make use of a Linux virtual server as a load balancer [9]. That is, we adopt a direct routing technique among the system structures supported by the Linux virtual server and use a round robin scheduling algorithm. When the Linux virtual server runs without an error, the LB probe generates 'LB_ALIVE' event. The LB handler maintains the service status table and it performs a procedure according to the event. In case the Linux virtual server has a failure, the LB probe removes the failed server perceived by CM from the available server list and makes the Linux virtual server restart.

NM(Node Manager): The NM is a component which manages a network communication with CM in the master node. It transmits both the event generated by each probe and the monitored information to the CM in the master node. It also perceives the status of CM according to the response of the CM. If the NM perceives the error of master node during the communication with the CM, the NM makes a connection with the backup node and transmits all the information to it. The NM also generates an event and transmit it to the CM when it perceives an error of each probe. If the NM does not receive a response from the CM during a defined interval, the NM considers that the CM has failed and makes a connection with a backup node.

CM(Cluster Manager): The CM running on the master node manages a service status table that describes the process status of each service, probe, and NM. The CM also perceives the status of networks by sending a ping message to each server node through both the cluster network and the service network. Based on the status of networks and service status table, the CM manages all system resources and services. It also analyzes the events received from the each server node and transmits them to a handler to perform the appropriate procedure. If the CM perceives an error of the NM, it makes the NM restart. If the CM perceives an error of network by using the ping message, it generates an event and transmits it to the corresponding service handler to perform its recovery procedure. Because the CM is running on the master node, the failure of the master node causes the failure of whole cluster-based DBMS. To solve the problem, the CM selects a backup node that plays a role of the master node when the master node has failed. The backup CM running on a backup node communicates with the CM of the master node and stores into its local disk all information which the master CM manages. If the CM perceives the failure the backup node, it selects one of the available server node as a backup node.

Recovery procedures for failures

Server nodes consisting of a cluster system can be classified into a master node, a backup node, and a database server node. Also the status of nodes can be classified into four types according to the status of both service and cluster networks. First, if there is no failure in both networks, the cluster system run with a normal situation. Secondly, if the service network fails, a server node can communicate with the other nodes, but it cannot receive a user request and return the result to the user. Thirdly, if the cluster network fails, a server node cannot communicate with the others and so a master node cannot distribute a user request to a server node. Finally, if both networks fail, the situation is considered as node failure since a node cannot work anymore. We can classify the status of node according to network failures as shown in Table 1.

Table 1. Classification of network failures

Status of node \ Network	Cluster network	Service network
Normal situation	O	O
Service network failure	O	X
Cluster network failure	X	O
Node failure	X	X

Master node failure: In the cluster system, a master node manages a cluster-based DBMS by distributing a user request to the each server node. The master node sends a ping message to each server node in the cluster system and perceive the failure of a node by analyzing the response of each server node. When the master node never receives the responses from all the server nodes, it regards the situation as the cluster network failure. Meanwhile, when the backup node cannot communicate with the master node using a ping message, it regards the situation as the master node failure. To prevent this situation, a backup node checks a failure of master node by sending it a ping message periodically and becomes a new master node when the master node has failed.

Backup node failure: In the cluster system, the backup node stores all the information received from the CM and monitors the master node. When the master node has failed, the backup node becomes a new master node and selects one of available server nodes as a backup node. When the backup node has failed, the cluster-based DBMS management tool perceives the failure and terminates the backup node. Then the backup node terminates active DB, Sysmon, NM, and its backup CM. In this time, the master node performs its recovery procedure to remove the backup node from available server nodes and to select a new backup node from them.

DB server node failure: In the cluster system, the failure of a server node can cause the failure of the entire cluster-based DBMS because the cluster-based DBMS uses a shared disk. Therefore, the cluster-based DBMS management tool should perform a recovery procedure to preserve data integrity by preventing the failed server node from accessing data. First, when the service network has failed, all the server nodes should stop its transactions and should perform its recovery procedure since it cannot return the result of a user request to the user. Secondly, when the cluster network has failed, all the server node should do the same thing as the above one because it cannot receive a user request from the master node as well as cannot communicate with the other server nodes. Finally, when the node has failed, the cluster-based DBMS management tool removes the server node from the available ones of the Linux virtual server and informs the other server nodes of the failure of the server node itself. In this time, the failed server node performs its recovery procedure to recovers its transactions and terminate database. The other server nodes should recover the data of the failed node.

User interface for our cluster-based DBMS management tool

We describe a convenient user interface for our cluster-based DBMS management tool that presents both the status of system information and the status of DBMS information that are monitored by a system probe on each server node. The system information consists of CPU, memory, disk, network, and node status. To present the status of CPU and memory, we use the file /proc/meminfo and /proc/stat. We depict the status of memory by the current usage rate of memory and the occurrence ratio of memory swapping. The left part of Figure 5 shows a user interface for the CPU and memory status of system information. In the figure, we show that our cluster system consists of four server nodes and the first node is the master node. In the master node, it is shown that the usage rate of CPU, memory, and swap memory are 25%, 96%, and 6%, respectively. To present the status of disk, we use the file /proc/mounts. We depict a current used disk space, an available disk space, and a total mounted space. The central part of Figure 5 shows a user interface for the disk status of system information. In the master node, it is shown that the total

mounted disk space, available disk space, and current used space is 4916Mbyte, 2502Mbyte, and 2413Mbyte, respectively. For the status of the network, we use the file /proc/net/dev. We depict the network information such as received packet, received error packet, sending packet, sending error packet. The right part of Figure 5 shows a user interface for the network status of the system information. In the master node, it is shown that the number of received packets, received error packets, sending packets, and sending error packets are 198, 0, 185, and 0, respectively. Here ACTIVE denotes that the corresponding service is available while INACTIVE denotes that the corresponding service is not running on the server node.

Figure 5. User interface for system information

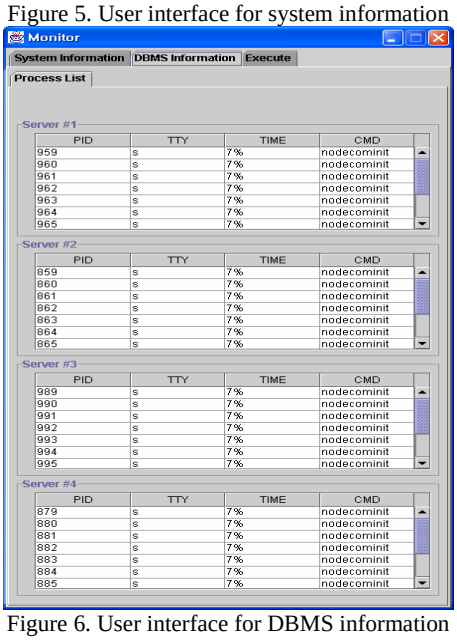

Figure 6. User interface for DBMS information

Figure 6 shows the user interface for DBMS information. We can recognize multiple DBMS processes and node commit is running on the master node. The process of PID 959 with S(Sleep) status makes use of 7% of CPU.

4. IMPLEMENTATION AND PERFORMANCE ANALYSIS OF CLUSTER-BASED DBMS MANAGEMENT TOOL

In this section, we implement our cluster management tool using iBASE/Cluster and do its testing and performance analysis. Table 2 describes our system implementation environment. For our implementation, we make use of Redhat Linux 7.1 operating system and iBASE/Cluseter DBMS [10]. Also, we make use of Linux virtual server as a load balancer [9].

Table 2. System implementation environment

System	450 Mhz CPU/HDD 30GB/128MB Memory * 4
OS	Redhat Linux 7.1 (Kernel 2.4.5)
Compiler	gcc 2.96 make 3.79.1
Database	iBASE/Cluster
Loadbalancer	linux virtual server 0.81

Testing for our cluster-based DBMS management tool

For the testing, we make use of a cluster system containing four server nodes each of which iBASE/Cluster is running on. In a normal situation, the service requests of users are transmitted to the master node. The Linux virtual server on the master node distributes a user request to one of the available server node according to its round-robin algorithm. Figure 7 shows the user interface of the normal situation where four server nodes containing the cluster system are active. The first server node serves as a master node and second server node serves as a backup node. In the figure, it is shown that the NM, the Sys (Sysmon) and the iBase (iBASE/Cluster) are in active state and the Linux virtual server and GLM (global lock manager) of the cluster-based DBMS is running on the master node. When the master node failure is occurred, we disconnect the master node from the cluster network that is used for the communication with other server nodes. Figure 8 shows the user interface of our cluster-based DBMS management tool when the master node has failed. In the figure, the second node functioning as the backup node plays a role of a new master node. It is shown in the first node that the NM, the Sysmon, the LB, and the iBASE/Cluster are not in active state. In addition, the Linux virtual server and GLM of the cluster-based DBMS is running on the second node. Meanwhile, when the backup node failure has occurred, we disconnect the backup node from the cluster network. The second server node functioning as a backup node is now in an inactive state and one of other server node is selected as a backup node.

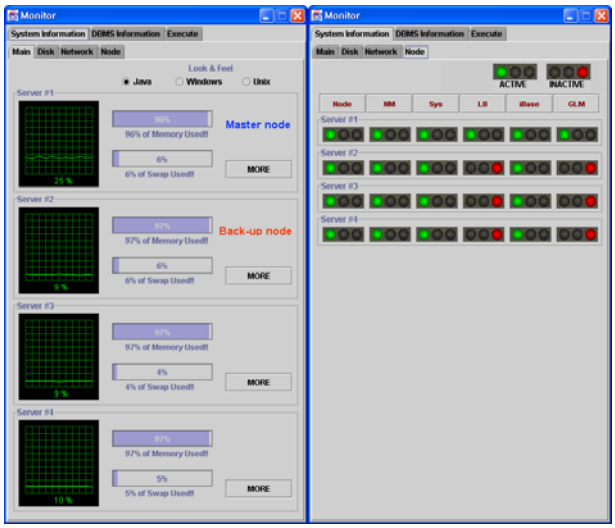

Figure 7 User interface of the normal situation

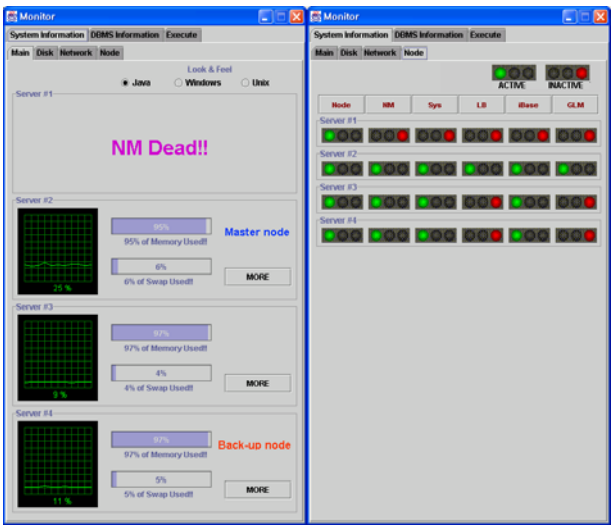

Figure 8 User interface when the master node failure has occurred

Performance analysis

We do the performance analysis of our cluster-based DBMS management tool using iBASE/Cluster. We cannot compare the performance of our cluster-based DBMS management tool with that of OCMS because it is a cluster-based DBMS management tool for Oracle, but does not support the recovery procedure for the iBASE/Cluster. Therefore, we estimate both a sensing time for three types of node failures and a time to perform a recovery procedure for them. Table 3 shows the sensing time and the recovering time for three types of node failures.

Table 3. Sensing time and recovering time for three types of node failures.

	Sensing time	Recovering time
Master node failure	0.91	0.78
Backup node failure	0.89	0.51
Database server node failure	0.81	0.71

First, when the master node failure is occurred, the backup node becomes aware of the master node failure by using the result of ping message sent to the master node. We set the limit of response time for the ping message to 2 second. The time for sensing the master node failure is 0.91 second and the time for doing its recovery procedure is 0.78 second. If the backup node plays a role of the master node, it sets its virtual IP and creates its thread monitoring the network status of nodes in the cluster system. Secondly, when the backup node failure is occurred, the master node becomes aware of the failure and selects one of available server nodes as a backup node. The time for sensing the backup node failure is 0.89 second and the time for doing its recovery procedure is 0.51 second. A new backup node creates a thread to monitor the master node and receives the information of the service status table of the master node periodically. Finally, when the database server node failure is occurred, the master node perceives the failure and performs its recovery procedure. The time for sensing the database server node failure is 0.87 second and the time for doing its recovery procedure is 0.71 second. If a database server node failure has occurred, the master node removes the server node from the available server list in order that a user request is not transmitted into the sever node anymore.

Table 4 shows the major differences between our cluster DBMS management soft-ware and OCMS. First, the cluster manager of OCMS performs its task on each node and communicate each other through the private network. Assume that there are N server nodes, the total number of messages for communication in OCMS is (N-1)*N. However, our cluster DBMS management software only needs 2N messages. That is to say, OCMS has more network overhead than our cluster DBMS management soft-ware. Secondly, OCMS uses a quorum partition to recognize the failure of other nodes. If the number of nodes is increased, more nodes attempt to read the quorum partition, thus resulting in the degradation of recovery performance. However, our cluster DBMS management software keeps the history status information of every node to recognize failures. If a node detects multiple node failures based on the comparison between a current status and a history status, we consider it the failure of the detecting node since the multiple node failures are not assumed to occur at the same time. Be-cause this scheme is not affected by the number of nodes, our cluster DBMS manage-ment software is efficient than OCMS in terms of scalability.

Table 4. Differences between our cluster DBMS management software and OCMS

	Our cluster Management Software	OCMS
Network Communication	Centralized	Distributed
Failure Detection	History status information	Quorum partition

5. CONCLUSTION

In this paper, we designed and implemented a cluster-based DBMS management tool managing a cluster-based DBMS efficiently. Our cluster-based DBMS management tool monitored the system resources of all the server nodes and became aware of the failure of nodes. When the failure has occurred, our cluster-based DBMS management tool performed its recovery procedure in order to perform a normal service, regardless of the failure. Our cluster-based DBMS management tool enables users to recognize a single virtual system image and can provide them the status of all the nodes and resources by using a convenient graphic user interface (GUI). Finally, we performed the testing of our cluster management tool using the iBASE/Cluster-based DBMS. The testing result shows that our cluster-based DBMS management tool can supports nonstop service by performing its recovery procedure even though a node has failed.

6. REFERENCES

[1] Rajkumar Buyya , High Performance Cluster Computing Vol 1,2, Prentice Hall PTR, 1999.

[2] High Performance Communication, `http://www-csag.cs.uiuc.edu/projects/communication.html`.

[3] C. S. You, “Linux Clustering”, Communication of the Korea Information Science Society, Vol 18, No 2, pp33~39, 2000.

[4] J. Y. Choi, S. C. Whang, “Software Tool for Cluster”, Communication of the Korea Information Science Society, Vol 18, No 3. pp40~47, 2000.
[5] Gregory, F.Pfister, In Search of Clusters 2nd Edition, Prentcs-Hall, 1998.
[6] Linux Clustering, http://dpnm.postech.ac.kr/cluster/index.htm.
[7] Oracle Corporation, "Oracle 8i Administrator's Reference Release3(8.1.7) for Linux Intel", chapter 7, Oracle Cluster Management Software, 2000.
[8] Putchong Uthayopas, Jullawadee Maneesilp, Paricha Ingongnam,"SCMS: An Integrated Cluster Management Tool for Beowulf Cluster System", Proceedings of the International Conference on Parallel and Distributed Proceeding Techniques and Applications 2000 , Las Vegas, Nevada , USA , 26 ~ 28 June 2000.
[9] Linux Virtual Server, http://www.linuxvirtualserver.org.
[10] Hong-Yeon Kim, Ki-Sung Jin, June Kim, and Myung-Joon Kim, "iBASE/Cluster: Extending the BADA-IV for a Cluster Environment", Proceeding of the 18th Korea Information Processing Society Conference, Vol. 9, No. 2, pp. 1769-1772, Nov. 2002.