An Experimental Approach to Constructing and Enhancing a PC Cluster System for Scalable Network Services

Yi-Hsing Chang J. Wey Chen*

Department of Information Management
Southern Taiwan University
Yungkong, Tainan County, 710, Taiwan, R.O.C.

{yhchang, peterchen} @mail.stut.edu.tw

Received 15 January2008; Revised 30 April 2008; Accepted 1 May 2008

Abstract. Although clustering might not be a panacea for today's ills, it might help the organization that is trying to maximize some of its existing resources. In this paper, we experimentally present a small-scale personal computer based cluster system for network servers as a low-cost alternative to traditional high-performance computing systems. The enhanced cluster system for scalable network services (CSSNS) consists of the parallel virtual file system (PVFS), the Linux Virtual Server (LVS), the Director, and several high-end Pentium PCs connected by high-speed switched Ethernet networks for I/O nodes and cluster nodes. The PC cluster system constructed for scalable network services is characterized by its high performance, high availability, high scalable file system, low cost, load balance on cluster servers, and convenient managing/set-up for a broad range of applications. The transmission performance experiments for the entire system further validates and confirms that CSSNS is truly a laboratory-made high-performance, high reliability and scalability personal computer based cluster system for scalable network servers.

Keywords: cluster system, parallel virtual file system, Linux Virtual Server, scalable network service, scalable parallel file system, network file system, IP load-balancing technique, single-IP-image

1 Introduction

With the explosive growth of Internet and WWW services and the increasing availability of inexpensive yet powerful personal computers and servers, there has been an increasing trend toward personal computer cluster systems for small-scale network services. The reasons for this trend are the good performance price ratios these systems offer, the availability of these systems, and the broad range of applications suitable for these systems. More specifically, the following features are essential for hardware and software solutions to support scalable and highly available network services for any web application [1, 2, 3].

- Scalability—The ability to scale to large, rapidly growing user populations without a major overhaul.
- 24 x 7 availability--Even a short downtime may cause millions of dollars in damage to e-commerce companies. A web service should stay operational and responsive even in the face of partial hardware and software failures.
- Consistency — The network service system should strictly maintain the consistency of user data.
- Manageability--Management of the web service must be effective and operationally manageable regardless of how it was designed and/or configured.
- Cost-effectiveness-- The total system cost for building a cluster of workstations/PCs for certain types of applications and different workloads must be cost-effective. This cost is determined by various architectural and managerial parameters including memory hierarchy, the interconnection network configurations of the cluster, and the management costs [4].

It is challenging for a network service to achieve all of these properties, especially when it must manage large amounts of persistent states, as this state must remain available and consistent even if individual disks, processes, or processors crash. In this paper, we build a small-scale cluster-based network server system as a low-cost alternative to traditional high-performance computing systems. The design and development of our cluster system for scalable network services (CSSNS) is characterized by its high performance, high availability, high scalable file system, low cost, broad range of applications, and convenient managing/set-up. The parallel virtual file system (PVFS) [5, 6, 7, 8] is deployed in the system to provide a high performance and scalable parallel file system for PC clusters. The Linux Virtual Server (LVS) [3, 9, 10, 11], a scalable and highly available server built on a cluster of loosely coupled independent servers, also provides end users a single system view so they may operate in a smooth environment. Performance comparisons and analyses of the PVFS cluster and the

* Correspondence author

combined PVFS and LVS cluster systems using a modified weighted least-connection scheduling algorithm (MWLC) and Virtual Server via Direct Routing (VS/DR) load-balancing technique were conducted to report our discoveries and derive our conclusions. Lastly, the information from a comparative review on a few commercial and/or research products which uses different designs and implementations for single-IP-image are also presented in this paper.

2 Related Concept and Works

The work described in this paper is part of a larger work to build a high performance yet low cost PC-based cluster systems for network services. This section presents a brief overview of the network file system, cluster technology for network service, the Parallel Virtual File System (PVFS), and Linux Virtual Server (LVS).

2.1 Network File System

Cluster computing has recently emerged as a mainstream method for parallel computing in many application domains with Linux leading as the most popular operating system for clusters. As researchers continue to push the limits of the capabilities of clusters, new hardware and software have been developed to meet the cluster's needs of computing. In particular, hardware and software for message passing have matured tremendously since the beginning of Linux cluster computing. In many cases, cluster networks rival the networks of commercial parallel machines. These advances have broadened the range of problems that may be effectively solved by clusters. As a result, the cluster and parallel computing are one of the best solutions for high performance network file systems.

Network file systems can be divided roughly into three groups: commercial parallel file systems, distributed file systems, and research parallel file systems. The three groups are described as follows:

The first group comprises of commercial parallel file systems such as PFS for the Intel Paragon [12], PIOFS and GPFS for the IBM SP [13], HFS for the HP Exemplar [14], and XFS for the SGI Origin2000 [15]. These file systems provide high performance and functionality desired for I/O-intensive applications but are available only on the specific platforms on which the vendor has implemented them.

The second group comprises of distributed file systems such as NFS [16], AFS/Coda [17, 18], InterMezzo [19], and GFS [20]. These file systems are designed to provide distributed access to files from multiple client machines, and their consistency semantics and caching behavior are designed accordingly for such access. The types of workloads resulting from large parallel scientific applications usually do not mesh well with file systems designed for distributed access. Distributed file systems are not designed for high-bandwidth concurrent writes that parallel applications typically require.

The third group consists of some research projects in the areas of parallel I/O and parallel file systems, such as PIOUS [21], PPFS [22], Galley [23], and PVFS [7]. PIOUS focuses on viewing I/O from the viewpoint of transactions, PPFS research focuses on adaptive caching and pre-fetching, and Galley looks at disk-access optimization and alternative file organizations. These file systems may be freely available but are mostly research prototypes not intended for everyday use by others. The PVFS project is an effort to provide a parallel file system for PC clusters, which provides a global name space, striping of data across multiple I/O nodes, and multiple user interfaces.

2.2 Cluster Technology for Network Services

Clusters of servers, connected by a fast network, are emerging as a viable architecture for building highly scalable and available services. This type of loosely coupled architecture is more scalable, more cost effective, and more reliable than a tightly coupled multiprocessor system. However, a number of challenges must be addressed to make cluster technology an efficient architecture to support scalable services.

We can see that in client/server applications there are many ways to dispatch requests to a cluster of servers in the different levels. In general, these servers provide the same service and contain the same set of contents. The contents are either replicated on each server's local disk, shared on a network file system, or are served by a distributed file system. Request dispatching techniques can be classified into the following four groups: server-side Round-Robin DNS approach, client-side approach, server-side application-level scheduling approach, and server-side IP-level scheduling approaches. The four groups are described as follows: [3].

The first group is server-side Round-Robin DNS approach, which has some problem such as the caching nature of clients and hierarchical DNS system. It easily leads the system to a dynamic load imbalance state among the servers; thus, it is quite a challenge for a server to handle its peak load.

The second group is the client-side approach, which has some problems that are not client transparent. They require modification of client applications, so they cannot be applied to all TCP/IP services. Moreover, they will potentially increase network traffic by extra querying or probing.

The third group is the server-side application-level scheduling approach, which has some problems, too. This approach requires establishing two TCP connections for each request, one is between the client and the load balancer, the other is between the load balancer and the server; thus, the delay is high. The overhead of dealing with HTTP requests and replies in the application-level is high. Therefore, the application-level load balancer will be a new bottleneck when the number of server nodes increases.

The fourth group is server-side IP-level scheduling approaches, such as Berkeley's Magic Router [24], Cisco's Local Director [25], IBM's TCP router [26], Net Dispatcher [27], ONE-IP [28], and Linux Virtual Server (LVS) [11].

Although server-side IP-level scheduling approaches present many great commercial products for practical applications, LVS was constructed in this project to meet our two main objectives. The foremost is to provide a high performance and highly available cluster of servers platform for future research into Linux clusters. The second objective is to pick an architecture which meets our low-cost implementations with fast dispatching for scalable network services. Detailed discussion on the merits of these products will be provided in the late section where we examine the existing approaches to distributing client's requests for a single service to different servers using one shared cluster address for all servers in the cluster.

2.3 Linux Virtual Server

The Linux Virtual Server (LVS) [3, 9, 10, 11] is a Linux project developed by China's National Laboratory for Parallel and Distributed Processing. It is a scalable and highly available server built on a cluster of loosely coupled independent servers. The architecture of the cluster is transparent to clients outside the cluster. The client interacts with the cluster as if it is working under a unified, yet highly available and powerful server.

The LVS directs network connections to the different servers according to predefined scheduling algorithms in the kernel and makes parallel services of the cluster appear as a service on a single IP address. Client applications interact with the cluster as if it were a single high-performance and highly available server. The clients are not affected by interactions with the cluster and do not need modification. Transparent adding or removing of a node in the cluster achieves scalability and detecting node or daemon failures and reconfiguring the system appropriately provide high availability.

There are three IP load-balancing techniques for directors: Virtual Server via Network Address Translation (VS/NAT), Virtual Server via IP Tunneling (VS/TUN), and Virtual Server via Direct Routing (VS/DR)[3]. The director running the modified kernel acts as a load balancer of network connections from clients who know a single IP address for a service to a set of servers that actually performs the work. In general, real servers are identical, they run the same service and they have the same set of contents. The contents are replicated on each server's local disk, shared on a network file system, or served by a distributed file system. The data communication between a client's socket and a server's socket connection occurs regardless of whether a TCP or UDP protocol is used.

2.4 Parallel Virtual File System

The parallel virtual file system (PVFS) [5, 6, 7, 8] project was developed by the Parallel Architecture Research Laboratory (PARL) established by the National Aeronautics and Space Administration (NASA) Goddard Space Flight Center. It is an effort to provide high performance and scalable parallel file systems for PC clusters. PVFS is an open source and released under the Gnu's Not Unix (GNU) General Public License (GPL). It requires no special hardware or modifications to the kernel. PVFS provides four important capabilities, including: consistent file name space across the machine, transparent access for existing utilities, physical distribution of data across multiple disks in multiple cluster nodes, and high-performance user space access for applications. For a parallel file system to be easily used, it must provide a name space which is the same across the cluster and it must be accessible via the utilities to which we are all accustomed. PVFS file systems may be mounted on all nodes in the same directory simultaneously, allowing for all nodes to view and access all files on the PVFS file system under the same directory scheme. Once mounted, PVFS files and directories can be cooperatively operated with other familiar tools such as Unix tools.

3 System Architecture

The enhanced cluster system for scalable network services (CSSNS) consists of the parallel virtual file system (PVFS), the Linux Virtual Server (LVS), the Director, and several high-end Pentium PCs connected by high-speed switched Ethernet network for I/O nodes and cluster nodes. The architecture of the CSSNS combined with PVFS and LVS is depicted in Fig. 1.

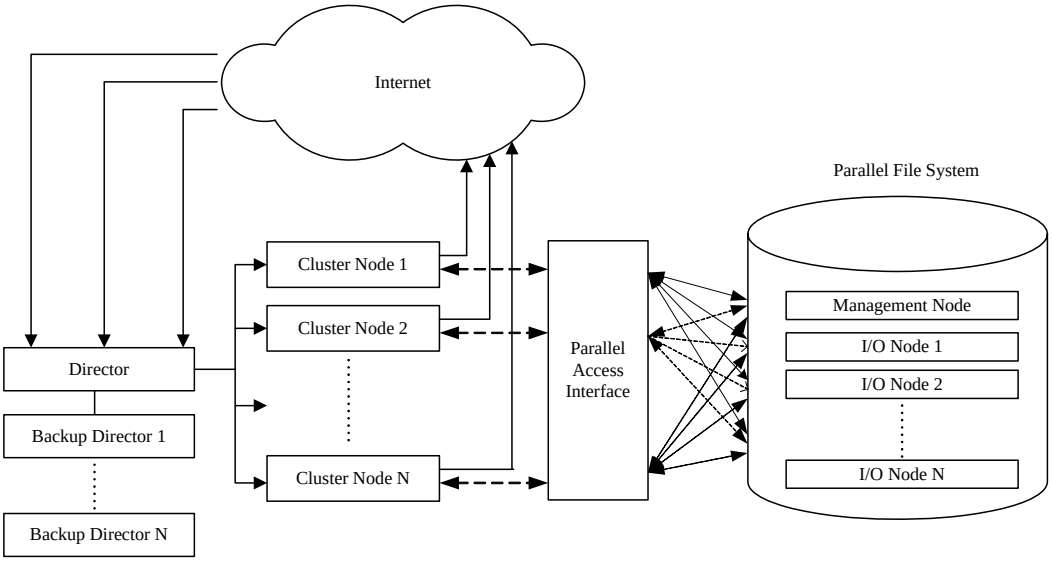

Fig. 1. The System Architecture of CSSNS

3.1 IP Load-balancing Technique

In Fig. 1, the VS/DR mode of the LVS is applied in our cluster system for network services. Although it is more complicated to configure. the environment for direct routing, it has the highest performance among all architectures of the LVS. Requests from the Internet flow to the director, and the director selects a cluster node for connection among many cluster nodes based on the scheduling algorithm used. The director keeps the session records by hashing tables between the clients and the chosen cluster node. Each cluster node receives exactly the same requests from the director as the clients from the Internet responds to the Internet clients directly without sending the responds back to the director. It is a highly efficient method to deal with requests from the Internet because the director does not have to process the responding packets from the cluster node. Each cluster node directly responds data to clients on the Internet.

To provide high-performance access to data stored on the system for many clients, the PVFS spreads data out across multiple cluster nodes, which is called I/O nodes. By spreading data across multiple I/O nodes, applications have multiple paths to data through the network and multiple disks to which data is stored. This eliminates single bottlenecks in the I/O path and increases the total potential bandwidth for multiple clients. For a low cost, high performance, and high scalability file system design in cluster nodes, we employ the PVFS for our parallel file system and the Linux kernel interface is used to access the parallel file system in PVFS.

3.2 Scheduling Algorithm Modification

There are four scheduling algorithms implemented in LVS: Round-Robin Scheduling, Weighted Round-Robin Scheduling, Least-Connection Scheduling, and Weighted Least-Connection Scheduling [24]. Because the weighted least-connection scheduling algorithm is dependent on the connections and processing capacities of the cluster nodes, it is perhaps a proper algorithm for the CSSNS to satisfy our high performance design criteria for network service. The administrator assigns a weight to each cluster node based on its relative processing speed, and network connections are scheduled to each cluster node by the percentage of the current number of live connections. Each cluster node is a ratio to its weight. A modified weighted least-connection scheduling algorithm (MWLC) is proposed in this project to optimize the scheduling process for the cluster system. Fig. 2 portrays the design to determine the next connected cluster node, where

M: the number of cluster nodes in CSSNS,
Wi: the weight of cluster node i, 1 <= i <= m,
Ci: the sum of file size of alive connections in cluster node i, 1 <= i <= m,
ALL_CONNECTIONS: $\sum$Ci, 1 <= i <= m.

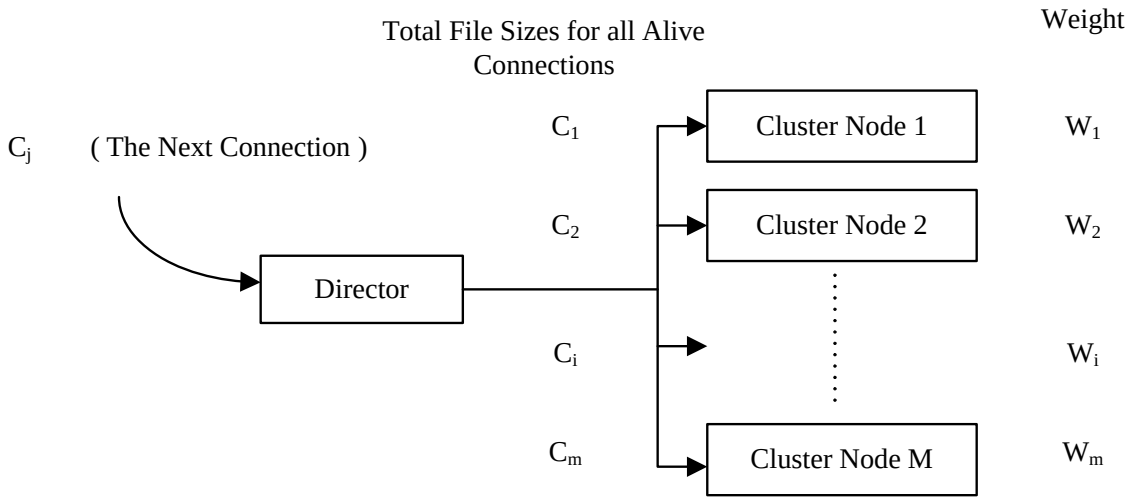

Fig. 2. Modified Weighted Least-Connection Scheduling

and the next network connection will be directed to the cluster node j, such that

$$(Cj/ALL\_CONNECTIONS)/Wj = \min \{ (Ci/ALL\_CONNECTIONS)/Wi \} \quad (1)$$

This formula can be simplified as

$$Cj/Wj = \min \{ Ci/Wi, i = 1 .. m \} \quad (2)$$

4 Experiment

4.1 Test Environment

As shown in Fig. 3, clients access network services from the director as requests are forwarded to the cluster nodes and the server processor receives data from parallel access interface then passes the responses directly to clients. The system comprises of seven personal computers: three of them are 1.7GHz Intel Pentium IV processors, 512MB of RAM, 100Mbps Ethernet Card, and each has a 40GB hard disk for cluster nodes, and the other four are for the director and I/O nodes with 1 GHz Pentium III processors with 256MB of RAM, 100Mbps Ethernet Card, and each has a 20GB hard disk. The operation system of all PCs is Redhat 6.2, which kernel version is 2.2.17. The package version of Parallel Virtual File System is 1.5.0, which kernel version is 0.9.0. The package version of Linux Virtual Server is 0.8.2. The networks are connected by DLINK DES-1016D 100Mbps full duplex locally and isolated from the Internet.

The network configuration includes three network segments. The first includes the connection that clients send requests to the director, and the connection that the cluster nodes respond data to the clients, which is called “Internet Connection”. The second network is the connection, which the director distributes requests to the cluster nodes, and is called “LVS connection”. The third is the connection for network file system, which is between the cluster nodes and the PVFS file system, and is called “PVFS connection”.

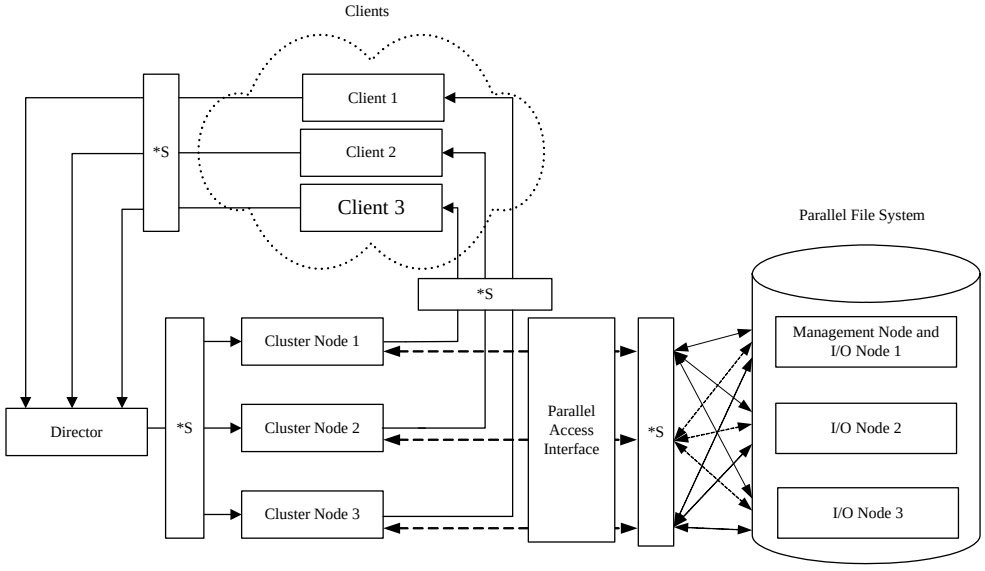

Fig. 3. The CSSNS Test Environment

4.2 Results

The performance of the two systems are compared, one is on a cluster node with PVFS and without LVS, and another system is on the combination with PVFS and LVS, which is tested on two and three cluster nodes, a director, and three I/O nodes. These systems are tested based upon different client numbers and file sizes. Requests produced by three clients, on average, are sent to the director simultaneously. The five results are illustrated as following.

Result 1. As shown in Table 1, Table 2, and Table 3, the CSSNS costs less time than the system only with PVFS. Each cluster node sends data to the client respectively, and parallel access to the file from the PVFS file system. Each cluster node can more efficiently communicate with the clients and respond to data for its clients quickly.

Table 1. Complete Time (Only with PVFS)

FS\PN	100	150	200	250	300	350	400	450
10	156.9	252.3	341.8	463.5	596.9	749.2	937.6	1384.5
20	261.1	335.6	476.4	632.7	812.1	1135.8	1354.7	1912.3
30	473.2	698.7	975.4	1427.9	1886.5	2014.2	2173.5	2411.6
40	563.4	723.5	1205.3	1732.8	2057.6	2192.4	2401.9	2693.6
50	717.2	896.2	1237.5	1847.3	2219.5	2356.6	2631.0	2846.5

*FS: file size (MB), PN: process number (unit: second)

Table 2. Complete Time for Two Cluster Nodes(CSSNS)

FS\PN	100	150	200	250	300	350	400	450
10	95.6	161.7	206.8	315.6	386.1	512.6	627.8	910.3
20	157.8	216.3	324.2	394.7	529.3	687.5	874.9	1214.7
30	287.5	457.1	615.3	874.5	1135.4	1254.1	1338.4	1488.4
40	367.2	483.1	756.7	1097.2	1317.0	1375.9	1487.5	1589.6
50	445.7	563.1	724.4	1086.9	1394.9	1481.3	1591.7	1657.4

*FS: file size (MB), PN: process number (unit: second)

Table 3. Complete Time for Three Cluster Nodes (CSSNS)

FS\PN	100	150	200	250	300	350	400	450
10	72.6	110.8	164.5	237.7	289.3	368.4	501.9	713.2
20	103.7	153.1	223.5	290.9	385.3	539.2	579.4	919.2
30	265.3	352.7	534.3	784.6	953.9	1068.7	1224.6	1305.3
40	305.1	374.6	634.7	912.5	992.8	1137.9	1263.7	1407.6
50	389.6	467.9	653.9	1003.1	1105.4	1317.5	1385.2	1488.7

*FS: file size (MB), PN: process number (unit: second)

Table 4. Complete Time Ratio for Two Cluster Nodes (CSSNS / PVFS)

FS\PN	100	150	200	250	300	350	400	450
10	0.61	0.64	0.61	0.68	0.65	0.68	0.67	0.66
20	0.60	0.64	0.68	0.62	0.65	0.61	0.65	0.64
30	0.61	0.65	0.63	0.61	0.60	0.62	0.62	0.62
40	0.65	0.67	0.63	0.63	0.64	0.63	0.62	0.59
50	0.62	0.63	0.59	0.59	0.63	0.63	0.60	0.58

*FS: file size (MB), PN: process number

Table 5. Complete Time Ratio for Three Cluster Nodes (CSSNS / PVFS)

FS\PN	100	150	200	250	300	350	400	450
10	0.46	0.44	0.48	0.51	0.48	0.49	0.54	0.52
20	0.40	0.46	0.47	0.46	0.47	0.47	0.43	0.48
30	0.56	0.50	0.55	0.55	0.51	0.53	0.56	0.54
40	0.54	0.52	0.53	0.53	0.48	0.52	0.53	0.52
50	0.54	0.52	0.53	0.54	0.50	0.56	0.53	0.52

*FS: file size (MB), PN: process number

Result 2. As shown in Table 4 and Table 5, the average completion time with PVFS and LVS for two and three cluster nodes is about 63% and 51% of the time with only PVFS respectively. Therefore, the performance of the system with two and three cluster nodes is about 1.6 (1/0.63) and 2 (1/0.51) times the performance of the system with only PVFS, respectively. It is possible to have more than three cluster nodes to improve the CSSNS, but it depends on your system load and connections.

To find the speedup, some wording notations are defined as follows, where

P: the performance of the cluster system,
P1: the performance of the cluster system for a single cluster node,
N: the total number of cluster nodes,
N': the number of added cluster node(s).

From result 2, we obtain the following formulas:

$$N=1+N' \quad (3)$$

The performance of the entire cluster system with more than three cluster nodes, as derived from Table 4 and Table 5, can be represented and predicted by the following formulas:

$$P=(0.4N'+1.2) * P1 \quad (4)$$

Table 6. Saving Time for Two Cluster Nodes

FS\ PN	100	150	200	250	300	350	400	450
10	61.3	90.6	135.0	147.9	210.8	236.6	309.8	474.2
20	103.3	119.3	152.2	238.0	282.8	448.3	479.8	697.6
30	185.7	241.6	360.1	553.4	751.1	760.1	835.1	923.2
40	196.2	240.4	448.6	635.6	740.6	816.5	914.4	1104.0
50	271.5	333.1	513.1	760.4	824.6	875.3	1039.3	1189.1

*FS: file size (MB), PN: process number (unit: second)

Table 7. Saving Time for Three Cluster Nodes

FS\PN	100	150	200	250	300	350	400	450
10	84.3	141.5	177.3	225.8	307.6	380.8	435.7	671.3
20	157.4	182.5	252.9	341.8	426.8	596.6	775.3	993.1
30	207.9	346.0	441.1	643.3	932.6	945.5	948.9	1106.3
40	258.3	348.9	570.6	820.3	1064.8	1054.5	1138.2	1286.0
50	327.6	428.3	583.6	844.2	1114.1	1039.1	1245.8	1357.8

*FS: file size (MB), PN: process number (unit: second)

Result 3. As shown in Table 6 and Table 7, the saving time grows with client numbers and the total amount of the files. That is, it saves more time when both the number of service requests and the number of clients increase. Even though this graph encompasses only values of file capacity ranging from 10 to 50, the relative saving time growth rates are still evident (Fig. 4 and Fig. 5). These saving time behaviors are greatly contributed by the load balancer device from the LVS which distributes network connections among a pool of similar devices.

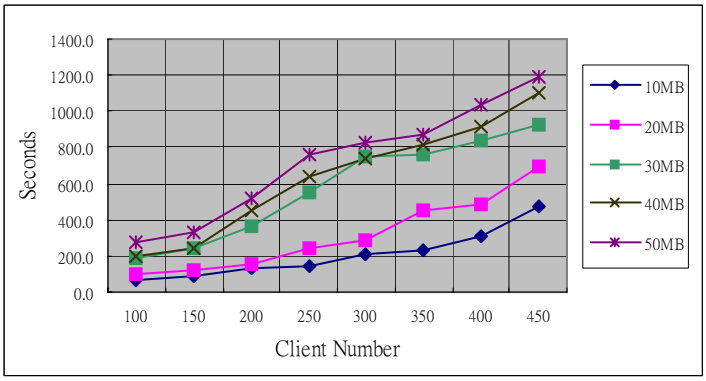

Fig. 4. Saving Time for Two Cluster Nodes

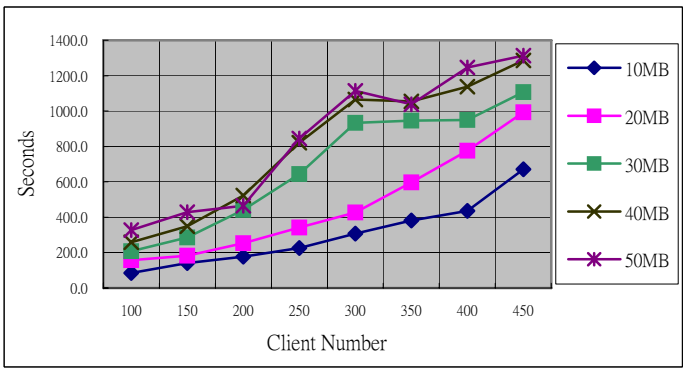

Fig. 5. Saving Time for Three Cluster Nodes

Result 4. As shown in Fig. 4 and Fig. 5, the saving time steadily grows with client numbers for moderately small amounts of file capacity (< 30 MB). Although the graphs for 30 MB, 40 MB, and 50 MB seem linear, it is easily verifiable that they are not by using a straight-edge. In essence, the performance of the cluster system is degraded when the client number exceeds three hundred. Two suggestions are provided to avoid this situation.

Suggestions 1: To increase the number of cluster nodes to share heavy workload. Since each cluster node has to construct one-to-many connections between I/O processes and I/O nodes, the processor of the cluster nodes must handle lots of connections, so the system performance slowdown is inevitable. Increasing the number of cluster nodes to share heavy workloads is one of the best ways for load balance.

Suggestion 2: To raise the network bandwidth. We have found that the network utility rate is usually full while the cluster node is busy dealing with large-scale files and plenty I/O processes. To improve the system performance, it must raise the network bandwidth to avoid the system performance abruptly shutting down.

4.3 Memory Load and PVFS Performance

Using the system monitor to observe the system load is a good way to understand the characteristics of system performance. The monitor shows that the system costs a lot of memory to run the program and the memory utility rate is full during the experiment. Measurement results for the memory utility rate are presented for a variety of testing parameters such as file size and the process number.

Fig. 6 illustrates the memory utility rate changes when the capacity of handled files vary with time. When the memory utility rate is 1, it means the memory load of the system is at normal state. When the rate value is greater than 1, it means the memory load of the system is getting worse than normal. If the value is 2, the memory load of the system is two times worse than the normal state and the system performance will degrade 50% from the normal operation.

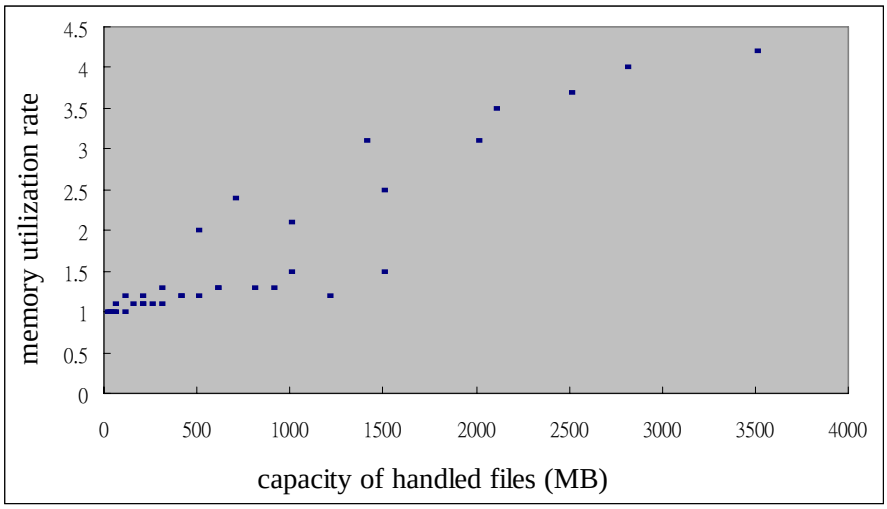

Fig. 6. Memory Load and Capacity of Handled Files

To find the formula, some notations are defined as follows. Let

P: process number,
S: file size in MB,
U: the memory utility rate,

According to Fig. 6, the following formula is obtained.

$$U = 0.001*P*S+0.9051 \quad (5)$$

From the formula, it shows that the relation of the memory utility rate, U, and the total capacity of handled files, P*S, is linear and under any operational circumstances we would like to keep the value of U equal to 1. When the value of U is equal to 1, the memory utility rate is at the normal operational state on cluster node. A memory utility rate of k means the system performance will degrade (1 – (1/k)) * 100% from normal level and k-

1 times the size of the current system memory on the compute mode must be added in order to bring the U value down to 1.

Suggestion 3: To optimize the performance of PVFS, the physical memory of the cluster node must be adjusted appropriately according to the calculated U value when the compute node has to handle many large-scale files. This can be done by adding memory to the system to bring the U value down to 1 (the normal operation level).

5 Discussions on Alternative Approaches for Single-IP-Image

The server-side single-IP-image approach implements a single IP image to the clients. Products of this category such as Berkeley's MagicRouter, Cisco's LocalDirector, IBM's TCP router and NetDispatcher, ONE-IP , and Linux Virtual Server (LVS) all have the design and different implementations, based on dispatching packets at the IP level, for providing single name images for a server cluster. They possess the advantages of fast dispatching, load-balancing, and ease of implementation and have become the defacto industry standard to distributing client's requests to different servers using only one published cluster address for all servers in the cluster.

Berkeley's MagicRouter [24] and Cisco's LocalDirector [25] use the Network Address Translation approach. In this approach, all address changes are performed by the server-site router, including changing the source addresses of the responding packets. First, the load balancer changes the destination address of request packets and forwards the selected server, then changes the source address of response packets back to the original address, so that clients believe packets are from a single IP address. However, the MagicRouter does not survive to be a useful system for others, the LocalDirector is too expensive, and they only support part of TCP protocol.

The TCP router [26] approach proposed by IBM achieves the single-address image by publicizing the address of the server-side router. It uses the modified Network Address Translation approach to build scalable web servers on IBM scalable Parallel SP-2 system. Every client request is sent to the router which then dispatches the request to an appropriate server based on load characteristics. The dispatching is performed by changing the destination IP address of each incoming IP packet to the address of a selected server. In order to create a seamless TCP connection, the selected server puts the router address instead of its own address as the source IP address in the replying packets. The advantage of the modified approach is that the TCP router avoids rewriting the reply packets and is totally transparent to the clients; the disadvantage is that it requires modification of the kernel code of every server in the cluster. NetDispatcher [27], the successor of TCP router, configures all server machines to have another IP address alias to the published cluster address in the set up, thus it no longer modifies the packet header or server machine kernels. The approach has good scalability, but NetDispatcher is a very expensive commercial product.

ONE-IP requires that all servers have their own IP addresses in a network and they are all configured with the same router address on the IP alias interfaces. ONE-IP supports two dispatching techniques, called routing-based dispatching which is based on a central dispatcher routing IP packets to different servers, and broadcast-based dispatching which is based on packet broadcasting and local filtering [28]. In both techniques, the destination server is selected by applying a hash function that maps the client IP address into a server identifier. The difference between the two techniques lies in the cluster component that applies the hash function. In routing-based dispatching, the Web switch selects the target server using the hash function, while in broadcast-based dispatching the Web switch broadcasts the packets destined to the Virtual IP address to every server in the cluster; each server evaluates whether it is the actual destination of the packets by applying the hash function [29]. The main advantage of the ONE-IP approach is that the Web switch does not need to keep track of any system state information and the rewriting of response packets can be avoided. The disadvantage is that it cannot be applied to all operating systems because some operating systems will shutdown the network interface when detecting IP address collision and the local filtering also requires modification of the kernel code of server [3]. Despite its merits, the absence of dynamic load balancing makes the ONE-IP approach less useful than other approaches. The goal of the ONE-IP project is solely designed to investigate the issues involved when the networking protocol stacks are modified to support a single-IP image for a cluster of machines.

6 Conclusion and Future Works

We constructed an enhanced cluster system for scalable network services, CSSNS, combined with PVFS and LVS. Each cluster node can be marked by its efficiency in communicating with the clients and its fast response. In order to sustain the system's performance, one could raise the network bandwidth to avoid performance deterioration. While PVFS will particularly benefit the most from parallel applications when bandwidth increases as multiple clients access data simultaneously [7], it may not always be cost-effective. A natural

solution for load balancing is to deploy a set of compute nodes. Increasing the I/O node may also improve the writing and reading performance of the PVFS. To optimize the performance of PVFS, the physical memory of cluster nodes must be adjusted appropriately according to the calculated U value when the compute node has to handle many large-scale files.

The performance test further revealed that the cluster system with fewer than three hundred clients is more efficient than one with over three hundred clients. Increasing the compute node to balance the connections with I/O nodes will improve the performance of PVFS. The results also revealed that parallel access with PVFS and LVS resulted in better performance and was more efficient than only with PVFS. Both LVS and PVFS are free software and are easy to be configured as a working component in the integrated environment for cluster network service. The overall system performance can be further improved by using powerful network devices such as Giga Ethernet.

PVFS, as a whole, is undergoing a full redesign at the current time. This will result in a complete rewrite of PVFS that incorporates new technology and lessons learned from the previous implementations. Some of the critical features that must be considered in the next generation are:

- Modular support for a variety of networking systems, so that the file system is no longer bound to TCP/IP but can take advantage of more advanced messaging protocols as they become available.
- Modular support for a variety of storage methods to allow I/O daemons to access local data through various methods, such as raw I/O or asynchronous I/O.
- Redundancy of both I/O data and metadata in case of system failure.

PVFS, in its current state, does not provide any redundancy or high security features. The existence of redundancy and security issues has prevented a scalable system such as CSSNS from widespread practical applications. However, the research is still going, and the authors believe that if PVFS were to provide access security, data redundancy and management node redundancy, then it would be more suitable for adoption as part of a highly scalable, reliable and fault-tolerant Linux cluster for general applications.

In the future, there are three important parts to be improved: scheduling algorithms for the director and PVFS, partitioning and stripping sizes of the file, and developing TCP redirector daemon inside the kernel. We may design a reactive scheduling algorithm that allows PVFS to adapt policies based on system state and application load in real time. The sizes of partitioning and stripping of the files for each network service will also be considered in design such a desirable and efficient scheduling algorithm. Much system flexibility can be obtained by porting/developing the TCP redirector daemon inside the kernel. By doing this, we can parse requests and do content-based scheduling, and we can explore higher degrees of fault-tolerance; transaction and logging process [27] would be tried to add in the load balance so that the load balancer can restart the request on another server and the client need not send the request again. All these efforts, when successfully completed and tested, will make Linux cluster system even more scalable, flexible, load-balancing, and fault-tolerant system for general network services.

7 Acknowledgement

This work was supported in part by the National Science Council in Taiwan under grant NSC92-2218-E-218-026.

References

[1] E. Brewer, “Clustering: Multiply and Conquer,” *Data Communications*, July 1997.

[2] W. Stallings, *Operating systems: Internals and Design Principles, 3rd edition*, Prentice-hall International, Upper saddle River, NJ, 1998.

[3] W. Zhang, S. Jin and Q. Wu, “Linux Virtual Server: Server Clustering for Scalable Network Services,” *Ottawa Linux Symposium*, 2000.

[4] X. Du, X. Zhang, and Z. Zhu, “Memory hierarchy considerations for cost-effective cluster computing,” *IEEE Transactions on Computers*, Vol. 49, No. 9, pp. 915-933, 2000.

[5] P. H. Carns, W. B. Ligon III, R. B. Ross and R. Thakur, “PVFS: A Parallel File System For Linux Clusters,” *Proceedings of the 4th Annual Linux Showcase and Conference*, pp. 317-327, 2000.

[6] P. C. Dibble, M. L. Scott, C. S. Ellis, ”Bridge: A High-Performance File System for Parallel Processors,” *Proceedings of Eighth International Conference on Distributed Computing System*, pp. 154-161, 1998.

[7] I. F. Haddad, “PVFS: A Parallel Virtual File System for Linux Clusters,” *Linux Journal*, pp. 74-82, 2000.

[8] Parallel Architecture Research Laboratory, http://parlweb.parl.clemson.edu/, Retrieved August 2007.

[9] Linux Virtual Server Cluster Configuration HOW TO, http://www.linuxdoc.org/, Retrieved August 2007.

[10] Linux Virtual Server project, http://www.linuxvirtualserver.org/, Retrieved August 2007.

[11] W. Zhang, S. Jin, and Q. Wu, *Creating Linux Virtual Server*, http://www.linuxvirtualserver.org/Joseph.Mack/linuxexpo99/linuxexpo2.html, Retrieved August 2007

[12] C. S. Freedman, J. Burger and D. J. DeWitt, “SPIFFI – A Scalable Parallel File System for the Intel Paragon,” In *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No 11, pp. 1185-1200, 1996.

[13] P. F. Corbett, D. G. Feitelson, J. P. Prost, G. S. Almasi, S. J. Baylor, A. S. Bolmarcich, Y. Hsu, J. Satran, M. Snir, R. Colao, B. Herr, J. Kavaky, T. R. Morgan and A. Zlotek, “Parallel file systems for the IBM SP computers,” *IBM Systems Journal*, pp. 222-248, 1995.

[14] R. Bordawekar, S. Landherr, D. Capps and M. Davis, “Experimental evaluation of the Hewlett-Packard Exemplar file system,” *ACM SIGMETRICS Performance Evaluation Review*, pp. 21-28, 1997.

[15] “XFS: A next generation journal led 64-bit file system with guaranteed rate I/O,” http://oss.sgi.com/projects/xfs/, Retrieved August 2007.

[16] B. Pawlowski, C. Juszczak, P. Staubach, C. Smith, D. Lebel and D. Hitz, “NFS Version 3 Design and Implementation,” *Proceedings of Summer USENIX Conference*, pp. 137-151, 1994.

[17] P. J. Braam., “The Coda distributed file system,” *Linux Journal*, Volume 50, 1998.

[18] The Coda project, CMU Coda Team, 1987-now, http://www.coda.cs.cmu.edu/, Retrieved August 2007.

[19] J. Braam, M. Callahan and P. Schwan, “The InterMezzo file system,” *Proceedings of the O’Reilly Perl Conference*, 1999.

[20] K. W. Preslan, A. P. Barry, J. E. Brassow, G. M. Erickson, E. N., C. J. Sabol, S. R. Soltis, D. C. Teigland, M. T. O'Keefe, “A 64-bit, shared disk file system for Linux”, *Proceedings of the Seventh NASA Goddard Conference on Mass Storage Systems*, 1999.

[21] S. A. Moyer and V. S. Sunderam, “PIOUS: a scalable parallel I/O system for distributed computing environments,” *Proceedings of the Scalable High-Performance Computing Conference*, pp. 71-78, 1994.

[22] J. Huber, C. L. Elford, D. A. Reed, A. A. Chien, and D. S. Blumenthal, “PPFS: A high performance portable parallel file system,” *Proceedings of the 9th ACM International Conference on Supercomputing*, pp. 385-394, 1995.

[23] N. Nieuwejaar and D. Kotz., ”The Galley parallel file system,” *Parallel Computing*, Vol.23, No.4, pp. 447-476, 1997,.

[24] E. Anderson, D. Patterson, and E. Brewer, “The magicrouter: an application of fast packet interposing,” http://www.cs.berkeley.edu/eanders/magicrouter, Retrieved December 2007.

[25] Cisco local director. Cisco Systems, Inc., http://www.cisco.com/warp/public/cc/pd/cxsr/400/index.html, Retrieved August 20, 2007.

[26] D. Dias, W. Kish, R. Mukherjee, and R. Tewari, “A scalable and highly available server,” *Proceedings of COMPCON 1996*, pp. 85-92, 1996.

[27] G. Goldszmidt and G. H. Netdispatcher, *A tcp connection router*, http://www.ics.raleigh.ibm.com/netdispatch/, Retrieved December 2007.

[28] O. P. Damani, P. E. Chung, Y. Huang, C. Kintala, and Y.M. Wang, “One-IP: Techniques for hosting a service on a cluster of machines.” *Computer Networks*, Vol.29, pp. 8-13, 1997.

[29] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu, “The state of the art in locally distributed web-server systems,” *ACM Computing Surveys*, Vol. 34, No. 2, pp. 263-311, 2002.