NAS Switch: A Novel CIFS Server Virtualization

Wataru Katsurashima, Satoshi Yamakawa,
Takashi Torii, Jun Ishikawa,
and Yoshihide Kikuchi
Internet Systems Research Laboratories,
NEC Corporation
w-katsurashima@bq.jp.nec.com,
s-yamakawa@cj.jp.nec.com,
t-torii@ce.jp.nec.com,
j-ishikawa@bc.jp.nec.com,
y-kikuchi@dc.jp.nec.com

Kouji Yamaguti, Kazuaki Fujii,
and Toshihiro Nakashima

NEC Software Kyushu, Ltd.

k-yamaguchi@px.jp.nec.com,
k-fujii@pv.jp.nec.com,
t-nakashima@pq.jp.nec.com

Abstract

This paper proposes a CIFS Server virtualization method which requires no proprietary software or hardware for clients or NAS units. The method is implemented as an in-band network application between clients and NAS units, and it provides users and administrators with a single virtual NAS system that incorporates all their units. Since almost all name resolution operations are performed by individual NAS units independently, use of this method imposes only a very light computational load and creates little latency.

1. Introduction

The ease of introduction and management of those file server appliances known as Network Attached Storage (NAS) devices has led to their recent extensive use in environments ranging from small offices to data centers.

The convenience of a NAS system tends to decrease gradually, however, as the number of NAS units in it increases because, while management becomes more complex in proportion to the number of units, the capacity and performance of such a storage system do not increase to the same degree. Management of multiple storage units can be particularly complex because, when data is being transferred from one unit to another, service for all clients must be stopped with respect to those units.

The creation of a single “virtual-NAS” system incorporating multiple storage units has been proposed to ease the problems of increased scale. Conventional approaches to such virtualization may be classified into two types, NAS-based virtualization [1, 2, 3] and out-of-band virtualization [4, 5]. Because both these approaches require that clients and/or NAS units be equipped with either proprietary software or hardware, however, they would be difficult to introduce and could not be efficiently adopted within existing systems.

By way of contrast, this paper proposes a virtualization method which requires no proprietary software or hardware for clients or NAS units. The method, described in the sections which follow, can also easily be generalized for use with common file servers. It is implemented as an in-band network application between clients and NAS units, and it provides users and administrators with a single virtual NAS system that incorporates all their units. Since almost all name resolution operations are performed by individual NAS units independently, use of this method imposes only a very light computational load and creates little latency.

While the use of the method with respect to the Network File System (NFS) is briefly discussed in Section 6, this paper otherwise focuses on its use with respect to the Common Internet File System (CIFS) protocol alone.

2. System Architecture

Figure 1 shows one possible configuration for the proposed method. The NAS Switch, our proposed virtualization switch, is simply connected to the IP network to which clients and NAS units are connected. Neither clients nor NAS units require any proprietary software or hardware; the only requirement is use of the CIFS protocol.

The NAS Switch provides clients with virtual share-folders, corresponding to the actual share-folders in the NAS units. It monitors CIFS packets and forwards each to its appropriate actual NAS share-folder. In some cases,

the NAS Switch may be able to provide responses by itself to client requests in CIFS packets without using an NAS unit (see Section 4 for details).

Note that we further assume here that clients also employ the Distributed File System (DFS) [6], but this assumption is safe for almost all current environments, including all Windows OSs beginning with Windows 98.

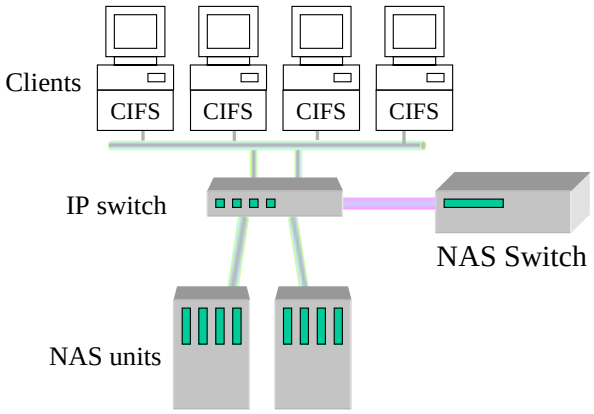

Figure 1. System Architecture

3. Functions

The NAS Switch hides the physical configuration of NAS units and provides a single virtual NAS system for users and administrators. It provides two main functions: (1) it enables administrators to link name spaces in NAS units hierarchically, to integrate them into larger "virtual name spaces", and to show these to users; and (2) it allows administrators to perform administrative data transfers between NAS units in a manner which is transparent (i.e. unseen) with respect to users.

3.1. Integration of name spaces

A virtual name space is composed of hierarchically-linked NAS name spaces (Figure 2). Since a virtual name space is independent of the actual NAS configuration, administrators can add a new NAS for increased capacity without changing the name spaces that users see.

3.2. Administrative Data Transfers between NAS units

It is easy for administrators to conduct data transfers in order, for example, to maintain a good balance in free-capacity among the various NAS units. Further, since such data transfers do not interrupt user operations in any way, they can be performed at any time on any data, even that which is open for operations by users.

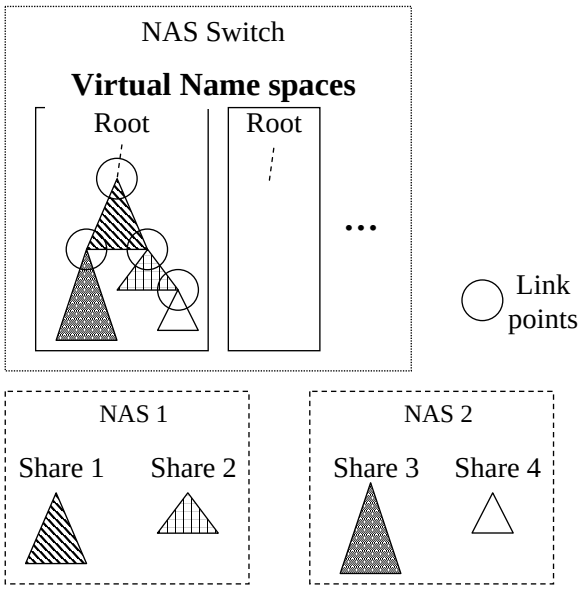

Figure 2. Virtual Name Space

4. Design and Implementation

Since in-band virtualization devices like the NAS Switch have an inherent potential to become a bottleneck, the first priority in designing them is to make their processing loads as light as possible. In this regard, the NAS Switch is designed to manage only junctions (i.e., link points) at which NAS name spaces are linked; all other name resolution operations are left to individual NAS units. This means a NAS Switch can forward almost all requests at high speed, resulting in low latency, as has been demonstrated by a Linux-based NAS Switch prototype which we have created (see Section 5 for details).

Figure 3 is a simple block diagram showing the composition of the NAS Switch. The response decision block receives requests from clients and determines those requests to which responses can be provided directly by the NAS Switch itself. Such requests are those which relate directly to *link points*. Specifically, they are

- requests across share-folders, and
- requests for referral.

Any other type of request is immediately forwarded to an appropriate NAS unit.

File I/O type requests (e.g., read or write requests) can be forwarded faster than other requests because any file access request can be classified roughly as either a file I/O type a or directory I/O type, and also because file I/O type requests do not relate to link points. The NAS Switch first sees the command number of a received request, and, if it is of the directory I/O type, the NAS

Switch determines whether or not it relates to link points by checking its pathname. If it is of the file I/O type, the NAS Switch immediately forwards the packet which include request header, that is, with the cut through mode.

The sequence which occurs when a client request crosses a *link point* is described below. If a request is across share-folders (i.e., is one which thus crosses a *link point*), the NAS switch returns a specific error to a client, in response to which the client returns a request for a referral, GET_DFS_REFERRAL. The NAS Switch receives this and returns a referral which gives information about the location of the requested resource. The client is then able to carry out a corrective redirection of the original request. The referral locates another virtual share-folder in the NAS Switch, and redirected requests are also received by the NAS Switch. Since a client performs the above exchanges transparently (i.e. unseen by its users), users are still able to browse virtual name spaces seamlessly. The mechanism for redirection is the same as that used in DFS, and here the NAS switch behaves the same as a DFS server.

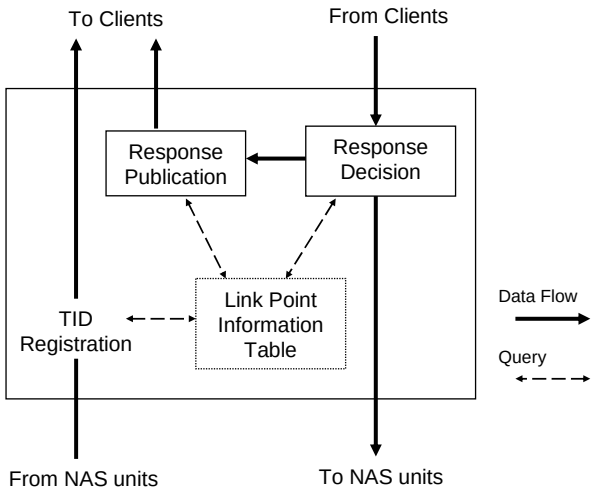

Figure 3. Composition of the NAS Switch

5. Performance Evaluation

This section presents the results of NAS Switch prototype experiments conducted to determine the overheads of the proposed system architecture. Here, we benchmarked systems both with/without the NAS Switch.

5.1. Experiment system

All experiments were run on the 1Gbps network shown in Figure 4. There were 4 clients, 2 NAS units and one NAS switch; all machines were linked in a Gigabit Ethernet network by means of 16-port Extreme Summit-5i switch. The manufacture of each machine is shown below.

NAS units: one NEC Corp. Express 5800 with dual 2.4-GHz Xeon processors, each having a 512KB L2 cache; 512MB of RAM; one 66Mhz, 64-bit PCI I/O bus; one 18GB Seagate Cheetah drive connected to an on-board SCSI controller; NTFS file system; a 3Com 3C996B-T 1000BaseT network controller, full duplex; Windows 2000 Server, Service Pack 3, with maximized throughput for file sharing.

NAS Switch: manufacture is the same as with the NAS units except for the OS, whose kernel is built from a Linux 2.4.17.

Clients: 4 clients are connected to form a single segment, and each of the clients is a 1.4-GHz Pentium III PC running a Windows 2000 Server with Service Pace 3, and having 1GB RAM and a 1000BaseT network card.

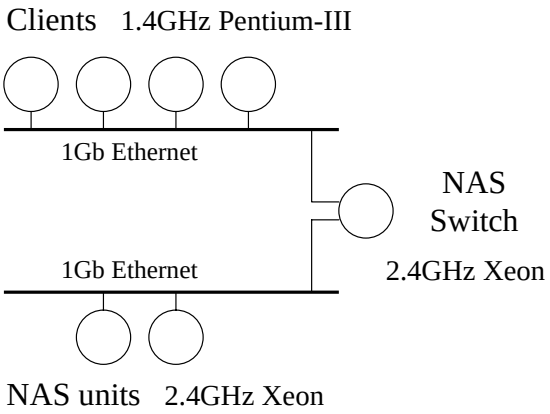

Figure 4. Test Network Topology

5.2. Experimental Results

Below are the results of latency comparison experiments conducted to determine whether or not the NAS Switch kept overhead sufficiently small.

5.2.1. Measurement method

We used eTesting Labs Inc.'s NetBench(R) version 7.0.2 with the Disk mix test script DM_NB702.SCR. Netbench is a benchmark program that measures how well a file server handles file I/O requests from 32-bit Windows clients, which pelt the server with requests for network file operations. Netbench reports throughput and client response time measurements.

In these experiments, clients mapped virtual share-folders (hosted by the NAS Switch) to network drives (e.g., F drives), and all requests from clients reached NAS units via the NAS Switch. Similarly, all responses from NAS units also reached clients via the NAS Switch. Two

clients accessed virtual share-folders mapped to share-folders hosted by one NAS unit, and the other two clients accessed virtual share-folders mapped to the other NAS unit.

Table 1. Overall results

	Average Response Time (milliseconds)	Elapsed Time (seconds)	Number of I/O Calls
Without use of the NAS Switch	*0.181*	*600.351*	*1896178*
With use of the NAS Switch	*0.307*	*600.089*	*1801632*

Table 2. Results for read/write I/O

Netbench's Commands (mapping Windows API)		Read	Write
	The percentages of the number	*39.7 %*	*31.1 %*
Average Response Time (milliseconds)	without use of the NAS Switch	*0.064*	*0.027*
	with use of the NAS Switch	*0.118*	*0.055*
	Differences	*0.054*	*0.028*

Table 3. Results for open/close I/O

Netbench's Commands (mapping Windows API)		Open File	Close
	The percentages of the number	*6.2 %*	*5.9 %*
Average Response Time (milliseconds)	without use of the NAS Switch	*0.224*	*0.254*
	with use of the NAS Switch	*0.462*	*0.576*
	Differences	*0.238*	*0.322*

5.2.2. Latency comparison

Table 1 shows a comparison of average response times for the system with/without use of the NAS Switch and demonstrate that the NAS Switch kept overhead very low: 0.12 milliseconds.

Table 2 shows a comparison of average response times for read/write I/O (file I/O type) requests. Since client caching worked very well, response times very low on the whole. The NAS Switch can forward file I/O type requests with the cut through mode, as mentioned above, and overheads for these requests were kept very too small. Since read/write I/O were big (almost all 64kB), there is a significant decrease in latency from input port to output port of the NAS Switch.

Table 3 shows a comparison of average response times for open/close I/O (directory I/O type) requests. Since the NAS Switch need check its pathname in order to determine, the overhead for this type of request was higher than that for a read/write request, it was still relatively low: 0.2 to 0.3 milliseconds.

These results show the design of the NAS Switch kept overheads small both file I/O type and directory type I/O.

6. Discussion

Let us consider here the potential of the proposed architecture for application to two particular uses.

Massive storage systems using the NAS Switch:

A critical requirement for most massive storage systems is huge throughput, and for such purposes, NAS Switches might be clustered. As noted earlier, the NAS Switch is designed to manage only junctions at which NAS name spaces are linked, and NAS Switches would have to share so little information among themselves that clustering would be relatively easy.

In even larger NAS systems, some degree of automatic load distribution would be needed, and for that purpose, we believe an appropriate approach might be policy-based load distribution, which would use administrative data migration between NAS units to eliminate system bottlenecks.

NFS (Network File System) Support:

Let us next consider potential support of the other standard file access protocol, the NFS protocol. While in this paper we have considered in detail only its support of CIFS, it should be noted that the NAS Switch can also provide a single virtual NAS system for NFS clients, managing only junctions at which NFS servers' name spaces are linked.

NFS clients would be handled slightly differently from CIFS clients. The NAS Switch would provide integrated exports to NFS clients and hide junctions from them because, unlike CIFS clients, NFS clients would not have a referral function. For this reason, when receiving a request across NAS units (e.g. to rename a NAS-1 namespace file as a NAS-2 namespace file), the NAS Switch would have to execute the required action itself; the server would be unable to do that on its own.

7. Related work

7.1. Distributed File System

The CIFS protocol employs DFS [6] to integrate NAS name spaces. With DFS, administrators can unify NAS share-folders by making them sub-folders of a share-folder. These sub-folders serve clients as links to other share-folders and make it easier for users to browse seamlessly among linked name spaces.

With respect to its use with most NAS virtualization, however, DFS presents the following problems:

- The OSs of some NAS types are not capable of employing DFS.
- Administrative work is severely limited by the fact that data currently open for user operations cannot be transferred.
- Share folders which contain sub-folders for linking other share-folders are a highly restricted resource; there can be only one per server.

The NAS Switch reported here helps to overcome all of these problems.

7.2. NAS-based and Out-of-band Virtualization

To date, the two most common approaches to creating a single virtual-NAS system have been (1) NAS-based and (2) out-of-band.

NAS-based virtualization [1, 2, 3] introduces a cluster file system for use with NAS units so that both the capacity and performance of the system can be increased seamlessly. It is consequently necessary either employ NAS units which all have the same proprietary architecture, or to install proprietary software in all NAS units. Moreover, when the scale of the system is increased, overhead for communications between NAS units can no longer be ignored, and this limits the potential for scale increases.

In contrast to this, out-of-band virtualization [4, 5] uses a metadata server, as well as the redirectors contained in client OSs. The metadata server operates independently of the NAS units themselves. It has a map that leads from directory trees to location information in files, and it provides unified name spaces for use over the entire system. Redirectors ask the metadata server for file locations and conduct data access directly to storage servers. It is thereby possible to increase storage capacity seamlessly and independently of name spaces. It is also necessary, however, to install proprietary software in all clients, and to provide an additional network, such as a SAN (Storage Area Network).

As noted earlier, these drawbacks make it difficult, if not wholly impractical, to introduce such virtualization into actual systems.

With our in-band virtualization method, however, introduction is simple because it does not require clients and NAS units to be equipped with any proprietary hardware or software. That is, it is like a Layer 7 switch in that it requires no change to existing systems. Although such in-band virtualization devices have the potential to become bottlenecks, our NAS Switch avoids this problem by managing only those points which link NAS name spaces, thus requiring only limited processing.

8. Conclusions

In this paper, we have proposed a method for creating a single virtual-NAS system which requires no software or hardware modification either to clients or to NAS units. The proposed method is designed to be implemented as an in-band switch between clients and NAS units and to provide a single virtual NAS system for users and administrators. Since almost all name resolution operations are performed by individual NAS units independently, use of this method imposes only a very light computational load and creates little latency.

References

[1] 1Vision Software, Inc. vNAS, http://www.the1vision.com/products/vnas/

[2] Spinnaker Networks, Inc. SpinServer, http://www.spinnakernet.com/

[3] Zambeel, Inc. Aztera, http://www.zambeel.com/

[4] Anupam Bhide, et al, “File Virtualization with DirectNFS,” Proc. of the 10th NASA Goddard Conference on Mass Storage Systems and the 9th IEEE Symposium on Mass Storage Systems, pp. 43-58, Maryland, USA, 2002.

[5] EMC, Celerra HighRoad, http://www.emc.com/products/software/highroad.jsp

[6] Distributed File System, http://www.microsoft.com/windows2000/techinfo/howitworks/fileandprint/dfsnew.asp